



opsi Handbuch opsi-Version 4.0.7

Inhaltsverzeichnis

1	Copyright	1
2	Einführung	1
2.1	Für wen ist dieses Handbuch?	1
2.2	Konventionen zu Schrift und Grafiken	1
3	Überblick opsi	2
3.1	Erfahrung	2
3.2	Features von opsi	2
3.3	opsi Erweiterungen	2
3.4	Struktur	3
4	opsi-Management GUI: opsi-configed	5
4.1	Voraussetzungen und Aufruf	5
4.1.1	Logging des <i>opsi-configed</i>	6
4.1.2	Die Auswahl der Sprache	6
4.1.3	Benutzerdefinierte Startparameter mit Java Web Start	7
4.2	Login	7
4.3	Copy & Paste, Drag & Drop	8
4.4	Auswahl des Bedienungsmodus	8
4.5	Depotauswahl	8
4.6	Clientauswahl	10
4.6.1	Die Clientliste	12
4.6.2	Auswahl von Clients	13
4.7	Clientauswahl und hierarchische Gruppen im <i>Treeview</i>	15
4.7.1	Prinzipieller Aufbau	15
4.7.2	How to	16
4.8	Client-Bearbeitung	17
4.8.1	Install By Shutdown, Uefi-Boot und WAN-Konfiguration	17
4.8.2	WakeOnLan-Funktionalität mit versetzten Zeiten	18
4.8.3	Auslösen von opsiclientd-Events (Push-Installation)	19
4.8.4	Für WAN-Clients: Pakete-Cache löschen	19
4.8.5	<i>on_demand</i> Ereignis auslösen (Push Installation)	19
4.8.6	Nachrichten senden (<i>Starte Meldungsfenster</i>)	19
4.8.7	Sessioninfo der ausgewählten Clients	20
4.8.8	Herunterfahren / Reboot der ausgewählten Clients	20
4.8.9	Externe Remotecontrol-Werkzeuge für die ausgewählten Clients aufrufen	20
4.8.10	Clients entfernen, erstellen, umbenennen, umziehen	22

4.8.11	Localboot-Produkte zurücksetzen	23
4.9	Produktkonfiguration	23
4.10	Property-Tabellen mit Listen-Editierfenstern	25
4.11	Geheime Property-Werte	27
4.12	Netboot-Produkte	27
4.13	Hardwareinformationen	28
4.13.1	Automatisierte Treiberintegration	29
4.14	Software-Inventur	30
4.15	Logdateien: Logs von Client und Server	31
4.16	Produkt-Defaultproperties	32
4.17	Host-Parameter in der Client- und der Serverkonfiguration	32
4.17.1	Verwaltung von User-Rechten und -Rollen	34
4.18	Depotkonfiguration	36
4.19	Gruppenaktionen	36
4.20	Produktaktionen	37
4.21	Server-Konsole	37
4.21.1	Verbindungsdaten und Berechtigungen	38
4.21.2	SSH-Terminal	39
4.21.3	Vordefinierte Befehle mit Eingabemasken	39
4.21.4	Befehle definieren	40
5	opsi-server	42
5.1	Überblick	42
5.1.1	Installation und Inbetriebnahme	43
5.1.2	Samba Konfiguration	43
5.1.3	Der Daemon opsiconfd	43
5.1.4	Notwendige System-User und Gruppen	44
5.1.5	Notwendige Shares	44
5.1.6	opsi PAM Authentifizierung	45
5.1.7	Problem-Management	45
5.2	Hinweise zum Wechsel zu Samba 4	46
5.2.1	Die /etc/opsi/opsi.conf: pcpatch und opsifileadmins	46
5.2.2	Freigaben-Konfiguration	47
5.2.3	Zugriff auf die Freigaben: clientconfig.depot.user	48
5.3	opsi-Kommandozeilen-Werkzeuge und Prozesse	48
5.3.1	Werkzeug: <i>opsi-setup</i>	48
5.3.2	Werkzeug <i>opsi-package-manager</i> : opsi-Pakete (de-) installieren	50
5.3.3	Werkzeug: <i>opsi-product-updater</i>	52
	Konfigurierbare Repositories	52

	Konfigurierbare Aktionen	53
5.3.4	Werkzeuge: opsi-admin / <i>opsi config interface</i>	53
	Übersicht	53
	Typische Verwendung	55
5.3.5	Serverprozesse: opsiconfd und opsipxeconfd	56
	<i>opsiconfd</i> -Überwachung: opsiconfd info	56
5.3.6	Serverprozess: opsi-atftpd	57
5.4	Web service / API Methoden	58
5.4.1	Web service / API Methoden seit opsi 4.0	58
	Übersicht	58
	<i>host</i> (server und clients)	61
	<i>group</i> (Gruppen Verwaltung)	61
	<i>objectToGroup</i> (Gruppen Verwaltung)	62
	<i>product</i> (product meta data)	62
	<i>productProperty</i> (Definition der product properties)	63
	<i>productPropertyState</i> (Depot oder Client spezifische product property settings)	64
	<i>productDependency</i> (product Abhängigkeiten)	64
	<i>productOnClient</i> (client spezifische Informationen zu einem Produkt z.B. Installationsstatus)	65
	<i>productOnDepot</i> (depot spezifische Informationen zu einem Produkt)	65
	<i>config</i> (Verwaltung der Defaultwerte der Hostparameter)	66
	<i>configState</i> (Verwaltung der clientspezifischen Hostparameter)	66
	<i>auditHardwareOnHost</i> (Clientspezifische Hardware Informationen)	67
	<i>auditHardware</i> (Client unabhängige Hardware Informationen)	68
	<i>auditSoftwareOnClient</i> (Clientspezifische Software Informationen)	69
	<i>auditSoftware</i> (Client unahängige Software Informationen)	70
	<i>auditSoftwareToLicensePool</i> (Lizenzmanagement)	70
	<i>softwareLicenseToLicensePool</i> (Lizenzmanagement)	71
	<i>softwareLicense</i> (Lizenzmanagement)	71
	<i>licenseContract</i> (Lizenzmanagement)	72
	<i>licenseOnClient</i> (Lizenzmanagement)	72
	<i>licensePool</i> (Lizenzmanagement)	72
	Beispiel für die Änderung eines Keys in mehreren Objekten	73
	Spezielle Methoden	73
	<i>configState_getClientToDepotserver</i>	74
	Kommunikation mit Hosts	74
	Arbeit mit Logs	75
	Tutorial: Arbeit mit Gruppen	75
5.4.2	Aktions-orientierte Methoden	76
5.4.3	Backend-Erweiterungen	82

5.4.4	Zugriff auf die API	82
5.5	opsi-backup	82
5.5.1	Einführung	82
5.5.2	Vorbedingungen für ein Backup	83
5.5.3	Quick Start	83
5.5.4	Elementare Teile von opsi	83
	Opsi Konfiguration	83
	Opsi Backends	83
	Opsi Depotfiles	84
	Opsi Workbench	84
	Opsi Repository	84
	TFTP-Verzeichnis	85
5.5.5	Das opsi-backup Programm	85
	Ein Backup anlegen	85
	Backups archivieren	86
	Ein Backup verifizieren	87
	Ein Backup wiederherstellen	87
5.6	Datenhaltung von opsi (Backends)	88
5.6.1	file-Backend	88
5.6.2	mysql-Backend	88
	mysql-Backend für Inventarisierungsdaten (Übersicht und Datenstruktur)	88
	mysql-Backend für Konfigurationsdaten (Übersicht)	92
	Initialisierung des mysql-Backends	93
	Konfigurieren der MySQL-Datenbank zum Zugriff von außen	95
5.6.3	HostControl-Backend	95
5.6.4	HostControlSafe-Backend	95
5.6.5	Konvertierung zwischen Backends	96
5.6.6	Bootdateien	96
5.6.7	Absicherung der Shares über verschlüsselte Passwörter	96
5.7	Wichtige Dateien des opsi-servers	97
5.7.1	Allgemeine Konfigurationsdateien in /etc	97
	/etc/hosts	97
	/etc/group	97
	/etc/opsi/backends/	97
	/etc/opsi/backendManager/	97
	/etc/opsi/hwaudit/*	98
	/etc/opsi/opsi.conf	98
	/etc/opsi/modules	98
	/etc/opsi/opsiconfd.conf	98

	/etc/opsi/opsiconfd.pem	98
	/etc/opsi/opsipxeconfd.conf	98
	/etc/opsi/opsi-product-updater.conf	99
	/etc/opsi/version	99
	/etc/init.d/	99
5.7.2	Bootdateien	99
	Bootdateien in /tftpboot/linux	99
	Bootdateien in /tftpboot/linux/pxelinux.cfg	99
5.7.3	Dateien in /var/lib/opsi	99
	/var/lib/opsi/repository	99
	/var/lib/opsi/depot	99
	/var/lib/opsi/ntfs-images	99
	Weitere Verzeichnisse	100
5.7.4	Dateien des file Backends	100
	/etc/opsi/pkeys	100
	/etc/opsi/passwd	100
	Übersicht /var/lib/opsi	100
	Konfigurationsdateien im Detail	101
	./clientgroups.ini	101
	./config.ini	101
	./clients/<FQDN>.ini	101
	/var/lib/opsi/config/templates	101
	/var/lib/opsi/config/depots/	102
	Product control files in /var/lib/opsi/config/products/	102
	Inventarisierungsdateien /var/lib/opsi/audit	104
5.7.5	opsi Programme und Libraries	104
	Programme in /usr/bin	104
5.7.6	opsi-Logdateien	105
	/var/log/opsi/bootimage	105
	/var/log/opsi/clientconnect	106
	/var/log/opsi/instlog	106
	/var/log/opsi/opsiconfd	106
	/var/log/opsi/opsipxeconfd.log	106
	/var/log/opsi/package.log	106
	/var/log/opsi/opsi-product-updater.log	106
	tftp log in /var/log/syslog	106
	c:\opsi.org\log\opsi_loginblocker.log	107
	c:\opsi.org\log\opsiclientd.log	107
	c:\opsi.org\log\opsi-script.log	107
5.8	Upgrade Anleitungen für den opsi-server	107
5.9	Hinweise zur Dateistruktur des opsi-linux-bootimages unter UCS 4.X	107

6	opsi-client	108
6.1	opsi-client-agent	108
6.1.1	Überblick	108
6.1.2	Verzeichnisse des opsi-client-agent	108
6.1.3	Der Service: opscientd	109
	Installation	109
	opscientd	109
	opscientd notifier	110
	opsi-Loginblocker	111
	Event-Ablauf	111
	Konfiguration	115
	Konfiguration unterschiedlicher Events	115
	Proxysupport-Konfiguration	116
	Steuerung der Produkte die ausgeführt werden pro Event	116
	Konfiguration über die Konfigurationsdatei	117
	Konfiguration über den Webservice (Hostparameter)	124
	Logging	125
	opscientd infopage	126
	Fernsteuerung des opsi-client-agent	127
	Nachrichten per Popup senden	128
	<i>Push</i> -Installationen: Event <i>on demand</i> auslösen	128
	Sonstige Wartungsarbeiten (shutdown, reboot, ...)	128
6.1.4	Anpassen des opsi-client-agent an Corporate Identity (CI)	129
	Anzupassende Elemente: opsi-winst	129
	Anzupassende Elemente: opscientd	129
	Anzupassende Elemente: kioskclient	130
	Schutz Ihrer Änderungen vor Updates: Das custom Verzeichnis	131
6.1.5	Sperrung des Anwender Logins mittels opsi-Loginblocker	132
	opsi-Loginblocker unter NT5 (Win2k/WinXP)	132
	opsi-Loginblocker unter NT6 (Vista/Win7)	132
6.1.6	Nachträgliche Installation des opsi-client-agents	132
	Installation des opsi-client-agent in einem Master-Image oder als Exe	132
6.1.7	Das Systray Programm des opsi-client-agents	133
6.2	Registryeinträge	134
6.2.1	Registryeinträge des opscientd	134
	opsi.org/general	134
	opsi.org/shareinfo	134
6.2.2	Registryeinträge des opsi-winst	134
	opsi.org/winst	134

7	Security	135
7.1	Einführung	135
7.2	Informiert bleiben	135
7.3	Allgemeine Serversicherheit	135
7.4	Authentifizierung des Clients beim Server	135
7.5	Authentifizierung des Servers beim Client	136
7.5.1	Variante 1: verify_server_cert	136
7.5.2	Variante 2: verify_server_cert_by_ca	137
7.6	Authentifizierung beim controlserver des Client	137
7.7	Konfiguration eines Admin-Networks	137
7.8	Der user pepoch	138
7.9	Webservice-Zugriffsbeschränkungen	138
7.10	Root Passwort des bootimages ändern	139
8	opsi Produkte	139
8.1	Localboot-Produkte: Automatische Softwareverteilung mit opsi	139
8.1.1	opsi Standardprodukte	139
	<i>opsi-client-agent</i>	139
	<i>opsi-winst</i>	139
	javavm: Java Runtime Environment	139
	opsi-configed	139
	jedit	139
	swaudit + hwaudit: Produkte zur Hard- und Software-Inventarisierung	139
	opsi-template	140
	opsi-template-with-admin	140
	shutdownwanted	140
	opsi-script-test	140
	opsi-wim-capture	140
	opsi-winpe	140
	opsi-uefi-netboot	140
	opsi-set-win-uac	140
	opsi-setup-detector	140
	opsi-logviewer	141
	config-win10	141
	config-winbase	144
8.1.2	Beeinflussung der Installationsreihenfolge durch Prioritäten und Produktabhängigkeiten	144
	Algorithm1: Produktabhängigkeit vor Priorität (Default)	146
	Algorithm2: Produktpriorität vor Abhängigkeit	146
	Erstellung von Prioritäten und Produktabhängigkeiten	147

8.1.3	Einbindung eigener Software in die Softwareverteilung von opsi	147
8.2	Netboot Produkte	147
8.2.1	Parametrisierung vom Linux Installationsbootimage	147
8.2.2	Automatische Betriebssysteminstallation unattended	148
	Überblick	148
	Voraussetzungen	148
	PC-Client bootet vom Netz	149
	pxelinux wird geladen	149
	PC-Client bootet von CD	150
	Das Linux Installationsbootimage bereitet die Reinstallation vor	151
	Die Installation von Betriebssystem und opsi-client-agent	152
	Funktionsweise des patcha Programms	152
	Aufbau der Produkte zur unattended Installation	153
	Vereinfachte Treiberintegration in die automatische Windowsinstallation	153
8.2.3	Hinweise zu den NT6 Netbootprodukten (Win 7 bis Win 10)	153
8.2.4	mementest	157
8.2.5	hwinvent	158
8.2.6	wipedisk	158
8.3	Inventarisierung	158
8.3.1	Hardware Inventarisierung	158
8.3.2	Software Inventarisierung	161
8.4	opsi Abo Produkte	161
8.4.1	Initiale Bereitstellung von Paketen aus dem Abobereich	161
8.4.2	Aktualisierung der Abo-Pakete / Konfiguration des opsi-product-updater	161
8.4.3	Festlegung von Default-Properties	162
8.4.4	Update-Abo <i>MS-Hotfixes</i>	162
	misc mshotfix-uninstall	165
	misc dotnetfx	166
	misc dotnetfx-hotfix	166
	misc ms-ie11	167
	misc ms-optional-fixes	167
	misc silverlight	167
8.4.5	Update-Abo <i>MS-Office Hotfixes</i>	167
	Updates für MS Office 2010 32-bit international: office_2010_hotfix	168
	Updates für MS Office 2013 32-bit international: office_2013_hotfix	168
	Updates für MS Office 2016 32-bit international: office_2016_hotfix	168
8.4.6	Update-Abo <i>opsi Standardprodukte</i>	169
	Customizing der Pakete durch zentrale Konfigurationen	169
	Customizing der Pakete durch preinst/postinst-scripts	169

Adobe Acrobat Document Cloud Classic : <code>adobe.reader.dc.classic</code>	170
Adobe Acrobat Document Cloud Continuous : <code>adobe.reader.dc.continuous</code>	171
Adobe Flashplayer : <code>flashplayer</code>	172
Google Chromium for Business	174
Apache OpenOffice : <code>ooffice4</code>	175
LibreOffice The Document Foundation : <code>libreoffice</code>	176
Mozilla Firefox : <code>firefox</code>	176
Mozilla Thunderbird : <code>thunderbird</code>	178
Oracle Jre : <code>javavm</code>	178
9 opsi Erweiterungen	180
9.1 Freischaltung kostenpflichtiger Module	180
9.2 User-Rollen (via <code>opsi-configd</code>)	182
9.3 <i>opsi directory connector</i>	182
9.3.1 Einführung	182
9.3.2 Vorbedingungen für die opsi Erweiterung <i>opsi directory connector</i>	182
Allgemeine Anforderungen	182
Hardware-Anforderungen	182
Software-Anforderungen	182
9.3.3 Installation	183
9.3.4 Konfiguration	183
Directory-Einstellungen	183
Verbindung zu Univention Corporate Server	184
Verhaltens-Einstellungen	185
Mappings	185
Manuelle Zuordnung von Gruppennamen	186
opsi-Verbindungs-Einstellungen	186
9.3.5 Den Connector ausführen	187
Beispiel: wiederkehrende Verarbeitung mit <code>systemd</code>	187
Beispiel: wiederkehrende Verarbeitung als Cronjob	188
9.4 <i>opsi WIM Capture</i>	188
9.4.1 Vorbedingungen für die opsi Erweiterung <i>opsi wim capture</i>	188
9.4.2 Quick Info	189
9.4.3 Einführung	190
9.4.4 Abläufe Übersicht	190
9.4.5 Abläufe Details	191
9.4.6 Produkte	197
Hauptprodukt <code>opsi-wim-capture</code>	197
Target Produkte	198

9.4.7	Windows Installation von einem Targetprodukt aus	199
9.4.8	Hilfsprodukt opsi-wim-info	200
9.4.9	Bekannte Einschränkungen und Probleme	200
9.5	opsi Linux Support	200
9.5.1	Unterstützt als opsi-client: Linux	200
9.5.2	Vorbedingungen für den opsi Linux Support	202
9.5.3	opsi-linux-client-agent: 15 Freistarts	203
9.5.4	Einspielen der Produkte	203
9.5.5	Einführung	203
9.5.6	Linux Netboot Produkte v4.0.6 auf Basis des Distributionseigenen Installers	204
	Bereitstellung und der Installationsmedien auf dem Server	205
	Allgemeine Properties der opsi v4.0.6 Linux Netboot Produkte	206
	Die Produkte: debian7 , debian8 und ubuntu14-04, ubuntu16-04	207
	Das Produkt ucs41 und ucs42	208
	Einrichtung eines lokalen deb http Repository	209
	debian8	209
	ubuntu16-04	209
	ucs41	209
	ucs42	210
	Die Produkte sles11sp4, sles12, sles12sp1	211
	Die Produkte redhat70 und centos70	212
9.5.7	Linux Netboot Produkte v4.0.5 ohne Distributionseigenen Installer	212
	Allgemeine Properties der v4.0.5 Linux Netboot Produkte	213
	ubuntu	214
	debian	214
9.5.8	opsi-linux-client-agent	214
	opsi-linux-client-agent: Installation: service_setup.sh	216
	opsi-linux-client-agent: Installation: opsi-deploy-client-agent	216
	opsi-linux-client-agent: Installation: Durch die opsi netbootprodukte	217
	opsi-linux-client-agent: opsiclientd Konfiguration	217
	opsi-linux-client-agent: Pfade	218
	opsi-linux-client-agent: Known Bugs	218
9.5.9	Beispiel Scriptteile	219
9.5.10	Linux Localboot Produkte	222
	Das Produkt l-opsi-server	222
	l-os-postinst für v4.0.5 Netboot installationen	224
	l-desktop	224
	l-system-update	225
	l-swaudit	225

l-hwaudit	225
l-jedit	225
9.5.11 Inventarisierung	225
9.5.12 UEFI / GPT Unterstützung	225
9.5.13 Roadmap	225
9.5.14 Proxy für <i>.deb</i> -Pakete einrichten und verwenden	226
9.6 opsi mit UEFI / GPT	226
9.6.1 Netboot-Produkte mit uefi Unterstützung:	226
9.6.2 Vorbedingungen für das Arbeiten mit UEFI / GPT	228
9.6.3 Weitere Hinweise für die Verwendung des opsi-Moduls UEFI / GPT	228
9.6.4 Einführung	229
9.6.5 Was ist Uefi und was ist hier anders ?	229
9.6.6 Was ist anders an GPT	229
9.6.7 UEFI Boot	230
9.6.8 UEFI Netboot	230
9.6.9 Opsi Unterstützung für UEFI Netboot	230
9.6.10 Installation	231
9.6.11 Konfiguration des DHCP Servers	231
Beispiel aus einer Konfiguration eines Linux isc-dhcp-server	231
Beispiel für die Konfiguration auf einem Windows DHCP-Server 2012 R2	231
9.6.12 Konfiguration des opsipxeconfd	233
9.6.13 Alternative elilo.uefi	233
9.6.14 Kriterien für ein <i>gutes</i> BIOS	233
9.6.15 Technisches	233
Technisches zu UEFI	234
Technisches GPT	235
opsi UEFI/GPT Roadmap	236
9.7 <i>opsi local image</i>	236
9.7.1 Vorbedingungen für die opsi Erweiterung <i>opsi local image</i>	236
9.7.2 Einführung	237
9.7.3 Konzept	237
9.7.4 Technisches Konzept	238
9.7.5 Abläufe	239
Initiale Installation	239
Restore eines Images	240
Löschen eines Images	241
9.7.6 Update eines Images: Automatisierter Workflow	241
9.7.7 Die opsi-local-image Produkte	241
UEFI Kompatibilität	242

Netboot Produkt zur Partitionierung	242
Netboot Produkte zur Installation von Windows	243
Netboot Produkte zur Installation von Linux	244
Netboot Produkte zum Backup und Restore	245
Localboot Produkt zur Ablaufsteuerung	246
9.7.8 Erweiterte opsi Service Methoden	246
9.7.9 Backuppartition	247
9.7.10 Capture Images (WIM) erstellen und verteilen	247
Capture Images (WIM) Einführung	247
Capture Images (WIM) Komponenten	248
Capture Images (WIM) Abläufe	248
9.7.11 Windows Installation von einem Targetprodukt aus	248
9.7.12 Hilfsprodukt opsi-wim-info	249
9.7.13 Erstellen eines eigenen Ubuntu <i>Proxy</i>	250
9.8 opsi vhd reset	250
9.8.1 Vorbereitungen für die opsi Erweiterung 'opsi vhd reset'	250
9.8.2 Einführung	250
9.8.3 Abläufe	250
Initiale Installation	250
Schnelle Wiederherstellung	253
Update eines Images mit <i>opsi-vhd-auto-upgrade</i>	254
9.8.4 Die opsi-vhd Produkte	255
UEFI Kompatibilität	255
Das opsi Netboot Produkt <i>opsi-vhd-win10-x64</i> und seine Properties	255
Das opsi Localboot Produkt <i>opsi-vhd-control</i> und seine Properties	256
Das opsi Localboot Produkt <i>opsi-vhd-auto-upgrade</i> und seine Properties	256
Bekanntes Probleme und Einschränkungen	256
9.9 opsi-Lizenzmanagement	256
9.9.1 Vorbereitungen für die opsi-Lizenzmanagement-Erweiterung	256
9.9.2 Überblick	257
Funktion und Features	257
Übersicht Datenbankmodell	257
Aufruf der Lizenzmanagement-Funktionen im <i>opsi-configed</i>	258
9.9.3 Lizenzpools	258
Was ist ein Lizenzpool?	258
Verwaltung von Lizenzpools	259
Lizenzpools und opsi-Produkte	260
Lizenzpools und Windows-Software-IDs	260
9.9.4 Einrichten von Lizenzen	261

Aspekte des Lizenzkonzepts	261
Lizenzvertrag erfassen	262
Lizenzmodell konfigurieren	262
Abschicken der Daten	263
9.9.5 Lizenzierungen bearbeiten	263
Beispiel Downgrade-Option	264
9.9.6 Zuteilungen und Freigabe von Lizenzen	265
opsi-Service-Aufrufe zur Anforderung und Freigabe einer Lizenz	265
Winst-Skriptbefehle für die Anforderung und Freigabe von Lizenzen	265
Lizenzverträge	266
Manuelle Administration der Lizenznutzung	266
Erhaltung und Löschung der Lizenzenverwendungen	267
9.9.7 Abgleich mit der Software-Inventarisierung	268
9.9.8 Übersicht über den globalen Lizenzierungsstand	268
Fall Downgrade-Option	269
9.9.9 Service-Methoden zum Lizenzmanagement	269
9.9.10 Beispielprodukte und Templates	271
9.10 opsi WAN/VPN-Erweiterung	271
9.10.1 Vorbedingungen für die WAN/VPN-Erweiterung	271
9.10.2 Überblick über die WAN/VPN-Erweiterung	272
9.10.3 Caching von Produkten	273
Protokoll zum Zugriff auf ein opsi-depot	273
Die .files-Datei	274
Ablauf des Produkt-Cachings	274
Konfiguration des Produkt-Cachings	275
9.10.4 Caching von Konfigurationen	276
Das lokale <i>Client-Cache-Backend</i>	276
Ablauf der Synchronisation von Konfigurationen	276
Konfiguration des Config-Cachings	277
9.10.5 Empfohlene Konfiguration bei Verwendung der WAN/VPN-Erweiterung	277
Wahl des Protokolls für das Caching der <i>Produkte</i>	278
Prüfung der Server-Zertifikate	279
9.11 opsi-Nagios-Connector	279
9.11.1 Einführung	279
9.11.2 Vorbedingungen	280
Vorbedingungen bei opsi-Server und -Client	280
Vorbedingungen beim Nagios Server	280
9.11.3 Konzept	280
opsi-Webservice Erweiterung	280

opsi-client-agent Erweiterung	281
9.11.4 opsi-Checks	281
Hintergrund zum richtigen Verteilen der Checks	281
opsi-check-plugin	283
Check: opsi-Webservice	284
Check: opsi-Webservice pnp4nagios-Template	284
Check: opsi-check-diskusage	286
Check: opsi-client-status	287
Check: opsi-check-ProductStatus	288
Check: opsi-check-Depotsync	289
Check: Plugin über OpsiClientd checken	290
9.11.5 opsi Monitoring Konfiguration	291
opsi Monitoring User	291
opsi Nagios-Connector Konfigurationsverzeichnis	292
Nagios Template: opsitemplates.cfg	292
opsi Hostgroup: opsihostgroups.cfg	294
opsi Server: <full name of the server>.cfg	295
opsi Clients: clients/<full name of the client>.cfg	295
opsi Check-Kommandos Konfiguration: opsicommands.cfg	296
Kontakte: opsicontacts.cfg	297
Services: opsiservices.cfg	298
9.12 <i>opsi-clonezilla</i> (frei)	299
9.12.1 Vorbedingungen für die opsi Erweiterung <i>opsi-clonezilla</i>	299
9.12.2 Einführung	300
9.12.3 Konzept	300
9.12.4 Interaktive Abläufe	300
Interaktives savedisk im expert mode	302
Interaktives savedisk	304
Interaktives savepart	306
Interaktives restoredisk	307
Interaktives restorepart	309
9.12.5 Nichtinteraktive Abläufe	310
9.12.6 opsi-clonezilla properties	311
9.12.7 opsi-clonezilla known bugs	313
9.12.8 Clonezilla Kommando Referenz	313
Sichern oder Wiederherstellen von Images	313
disk-to-disk Operation	319
9.13 opsi-server mit mehreren Depots (frei)	320
9.13.1 Konzept	320

9.13.2	Erstellung und Konfiguration eines Depot-Servers	322
	Non-interaktive Registrierung eines opsi-depotserver	324
9.13.3	Paketmanagement auf mehreren Depots	325
9.14	Dynamische Depotzuweisung (frei)	325
9.14.1	Einführung	325
9.14.2	Voraussetzungen	326
9.14.3	Konfiguration	326
9.14.4	Editieren der Depoteigenschaften	327
9.14.5	Synchronisation der Depots	328
9.14.6	Ablauf	329
9.14.7	Template des Auswahlscripts	330
9.14.8	Logging	332
9.15	opsi Software On Demand (Kiosk-Mode) (frei)	333
9.15.1	Einführung	333
9.15.2	Vorbedingungen für das Modul	333
9.15.3	Konfiguration	334
	Produktgruppen pflegen	334
	Software-On-Demand-Modul konfigurieren	335
	Systemweite Konfiguration	335
	Client-spezifische Konfiguration	336
	opsiclientd Event-Konfiguration	337
9.15.4	Neue opsi Client Kiosk Anwendung	337
	Client Kiosk: Installation	338
	Client Kiosk: Verwendung	338
	Besonderheiten	342
	Client Kiosk: Anpassung an Corporate Identity	343
9.16	<i>User Profile Management</i> (frei)	343
9.16.1	Vorbedingungen für die opsi Erweiterung <i>User Profile Management</i>	343
9.16.2	Einführung	343
9.16.3	Konzept	344
9.16.4	Neue und erweiterte <i>opsi-winst</i> Funktionen	344
9.16.5	Beispiele von userLoginScripten	346
9.16.6	Konfiguration	350
9.16.7	Notification	350
9.17	opsi Installation beim Shutdown (frei)	351
9.17.1	Einführung	351
9.17.2	Vorbedingungen für die Installation beim Shutdown	351
9.17.3	Inbetriebnahme der Installation beim Shutdown	351
9.17.4	Technisches Konzept	352

Überblick	352
Durchführung per Shutdown-Skript	352
Registry-Einträge für die Ausführung des Shutdown-Skripts	353
Erforderliche Konfiguration des opsiagentd	354
Spezielle Konfiguration der Installation bei Shutdown	355
Lokale Logdatei für den Fehlerfall	356
9.18 opsi Feature <i>SilentInstall</i> (frei)	357
9.18.1 Vorbedingungen für die Silent Installation	358
9.18.2 Überblick über das SilentInstall-Feature	358
9.18.3 Auslösen der Silent Installation	358
9.18.4 Konfigurationen des opsi-Feature: <i>SilentInstall</i>	359
9.19 opsi Setup Detector (frei)	361
9.19.1 Einführung	361
9.19.2 Vorbedingungen für die Benutzung des opsi Setup Detectors	361
9.19.3 Inbetriebnahme des opsi Setup Detectors	361
Sprachunterstützung	362
Dateien des opsi Setup Detectors	362
9.19.4 Das Menü des opsi Setup Detektors	363
9.19.5 Automatische Analyse eines Setup-Paketes	363
9.19.6 Setup-EXE mit eingebettetem MSI	364
9.19.7 Unterstützte Installer-Typen	364
Installer-Typ MSI	364
Installer-Typ Advanced+MSI	365
Installer-Typ Inno Setup	366
Installer-Typ InstallShield	368
Installer-Typ InstallShield+MSI	369
Installer-Typ NSIS	371
9.19.8 Erzeugen eines neuen opsi Paketes	372

1 Copyright

Das Copyright an diesem Handbuch liegt bei der uib gmbh in Mainz.

Dieses Handbuch ist veröffentlicht unter der creative commons Lizenz *Namensnennung - Weitergabe unter gleichen Bedingungen* (by-sa).



Eine Beschreibung der Lizenz finden Sie hier:

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

Der rechtsverbindliche Text der Lizenz ist hier:

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

Die Software von opsi ist in weiten Teilen Open Source.

Nicht Open Source sind die Teile des Quellcodes, welche neue Erweiterungen enthalten die noch unter Kofinanzierung stehen, also noch nicht bezahlt sind.

siehe auch: <http://uib.de/de/opsi-erweiterungen/erweiterungen/>

Der restliche Quellcode ist veröffentlicht unter der GPLv3:



Der rechtsverbindliche Text der GPLv3 Lizenz ist hier:

<http://www.gnu.org/licenses/agpl-3.0-standalone.html>

Deutsche Infos zur GPLv3: <http://www.gnu.org/licenses/agpl-3.0.de.html>

Für Lizenzen zur Nutzung von opsi im Zusammenhang mit Closed Source Software kontaktieren Sie bitte die uib gmbh.

Die Namen *opsi*, *opsi.org*, *open pc server integration* und das opsi-logo sind eingetragene Marken der uib gmbh.

2 Einführung

2.1 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.

2.2 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 */var/lib/opsi/depot*.

Das Softwarepaket: *<opsi-depot-share>/office* liegt dann tatsächlich unter */var/lib/opsi/depot/office*.

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

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

3 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 Lizenzkosten einher geht, bietet opsi als Open-source-Werkzeug deutliche Kostenvorteile. Hier fallen nur die Kosten an, die von Ihnen durch tatsächlich angeforderte Dienstleistungen, wie Beratung, Schulung und Wartung, entstehen bzw. soweit kostenpflichtige Module benutzen wollen geringe Kofinanzierungsbeiträge.

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 Systemadministratoren durch einen erfahrenen Partner dringend geboten. Hier bietet Ihnen uib seine Dienstleistungen rund um opsi an.

Das von uib entwickelte System basiert auf 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.

3.1 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 auf der link:<https://www.opsi.org/opsi-map/>

3.2 Features von opsi

Die wesentlichen Features von opsi sind:

- automatische Softwareverteilung
- Automatische Betriebssysteminstallation
- Hard- und Softwareinventarisierung
- Komfortable Steuerung über das opsi Managementinterface
- Unterstützung von mehreren Standorten mit Depotservern

3.3 opsi Erweiterungen

- Lizenzmanagement
- MySQL-Backend
- Nagios Connector
- Installation bei Shutdown
- Local Image Backup (Lösung zur schnellen Wiederherstellung von Schulungscomputern (derzeit nur für die öffentliche Hand z.B. Schulen))
- Linux Agent
- WAN Erweiterung (Einbindung von Clients hinter langsamen Leitungen)
- User Profile Management: User Profile z.B. in einer Roamig-Profil Umgebung können modifiziert werden.
- OTRS::ITSM Connector (von unserem Partner Cape-IT)

3.4 Struktur

Die Konfiguration von opsi benötigt eine Datenhaltung. Die externen Komponenten von opsi kommunizieren mit dem opsi-server über einen Webservice. Der Prozess *opsiconfd*, welcher den Webservice bereitstellt, übergibt die Daten dem Backendmanager, der die Daten in das konfigurierte Backend schreibt.

Dabei unterstützt opsi unterschiedliche Backends:

- File-basiert
- MySQL-basiert

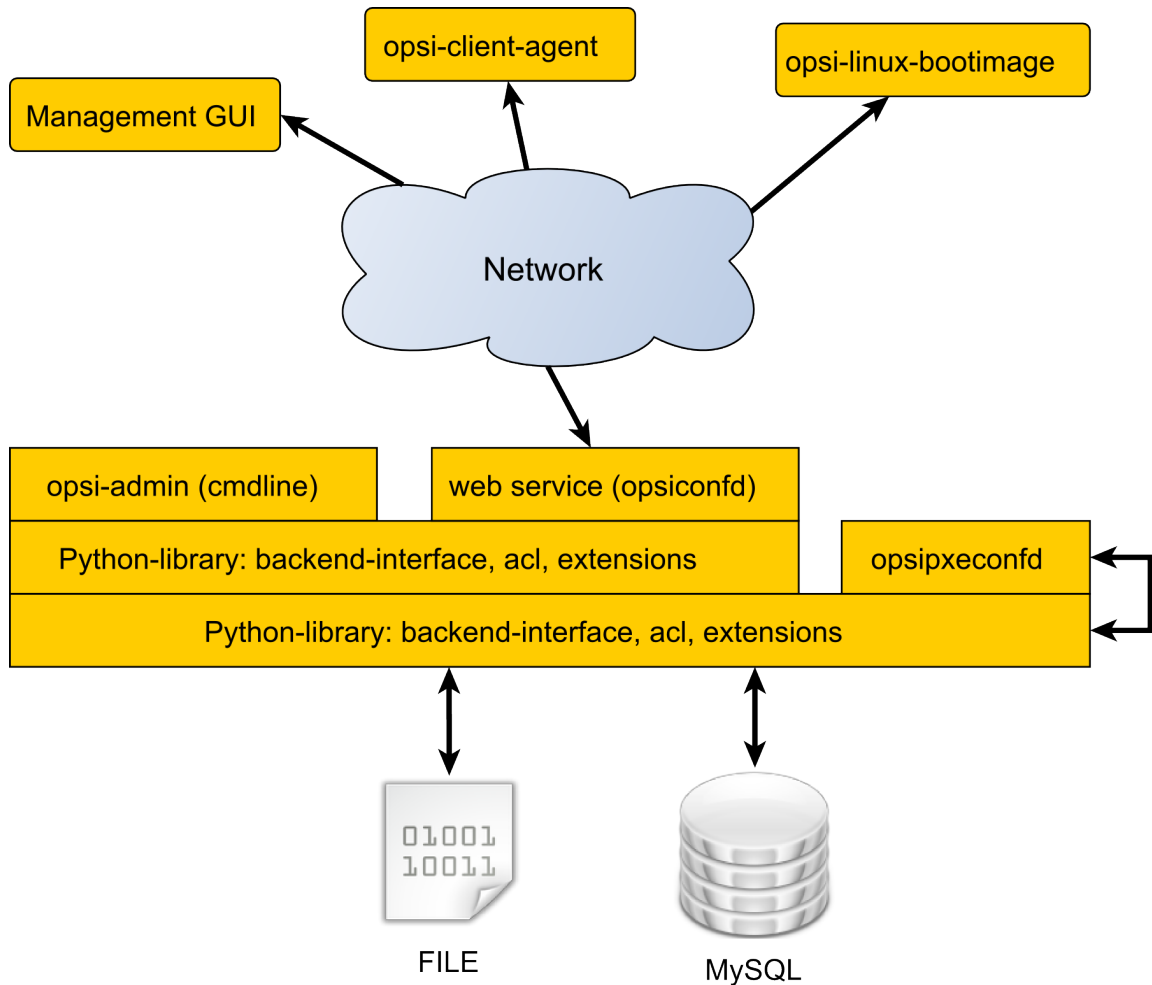


Abbildung 1: Schema: opsi mit File- und MySQL-Backend

Mehr zur Datenhaltung finden Sie im Abschnitt [5.6](#).

Die Konfiguration der Backends erfolgt in den Konfigurationsdateien in den Verzeichnissen `/etc/opsi/backendManager` sowie `/etc/opsi/backends`.

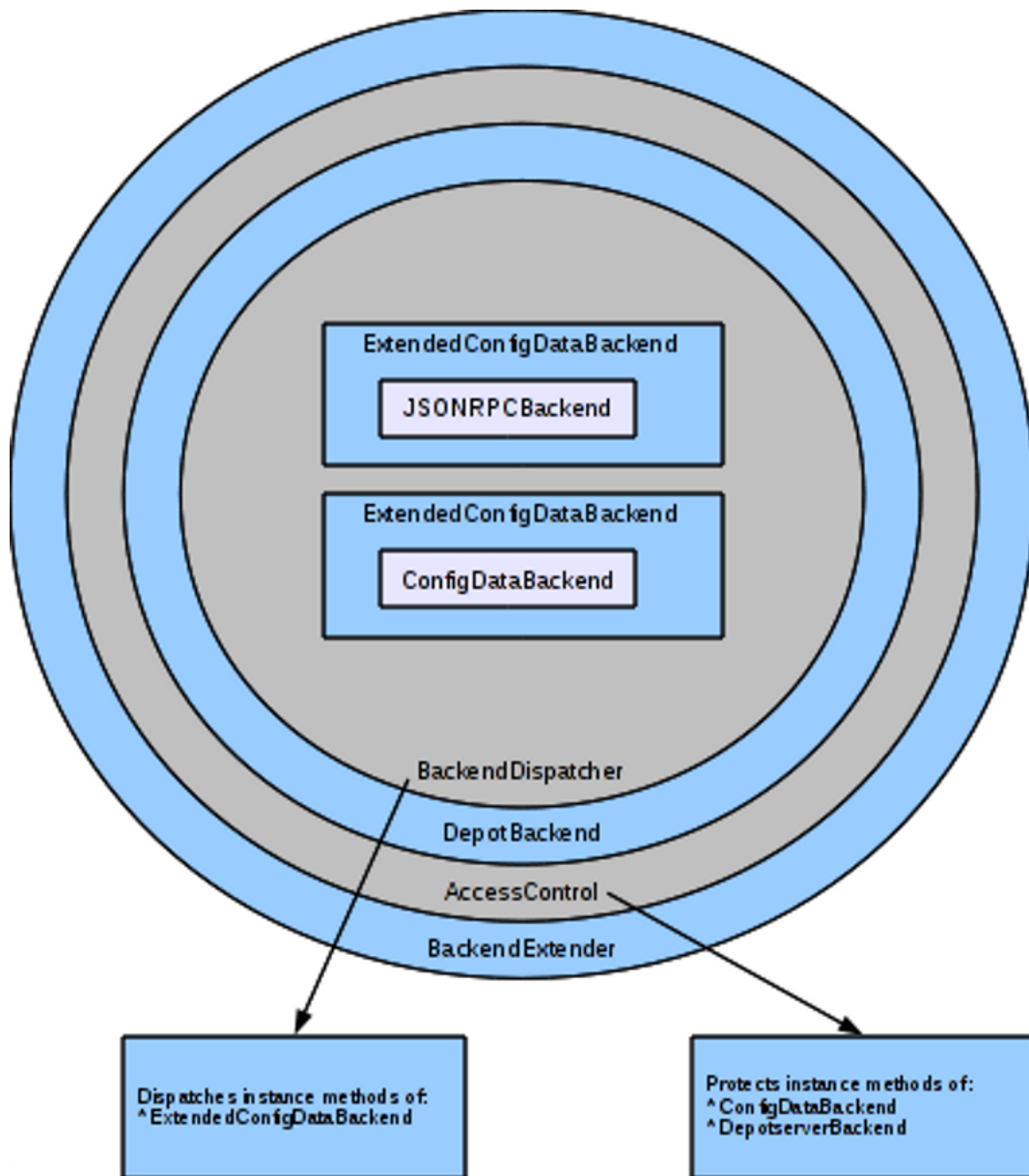


Abbildung 2: Schema: Schichten der Backend-Konfiguration

In den Konfigurationsdateien im Verzeichnis `/etc/opsi/backends` werden die Backends definiert.

Welche Backends für welche Zwecke verwendet werden, steht in der Datei `/etc/opsi/backendManager/dispatch.conf`.

Wer auf welche Methoden zugreifen darf, ist in der Datei `/etc/opsi/backendManager/acl.conf` konfiguriert.

Unterhalb von `/etc/opsi/backendManager/extend.d` können weitere opsi-Methoden definiert sein, welche die Basis opsi-Methoden verwenden.

So ist z.B. das Mapping der vorgangsbasierten Methoden (*Legacy*) auf die objekt-basierten Methoden in der Datei `/etc/opsi/backendManager/extend.d/20_legacy.conf` definiert. Genaue Beschreibungen dieser Konfigurationsdateien finden Sie im Abschnitt [5.7.1](#).

4 opsi-Management GUI: opsi-configed

4.1 Voraussetzungen und Aufruf



Wichtig

Ab Version 4.0.7.5.22 wird der *opsi-configed* in einer Fassung ausgeliefert, die mit Java 1.8 kompiliert ist und daher zum Ablauf eine Java-Laufzeitumgebung mindestens der Version 1.8 benötigt.

opsi-configed arbeitet mit den Daten eines auf dem Server laufenden *opsiconfd* mindestens der Version 4.0.6.

Das Programm steht auf download.uib.de in der jeweils aktuellen Version als opsi-Paket zur lokalen Installation zur Verfügung. Auf Nicht-opsi-Systemen kann der *opsi-configed* auch durch Kopieren der zum Paket gehörenden Dateien installiert werden. Zur Vereinfachung dieses Installationsvorgangs kann von <http://download.uib.de/opsi4.0/helper/> das Setupprogramm *opsi-configed-setup.exe* bezogen werden.

Die jeweils stabile Version des *opsi-configed* ist auch Element der opsi-Server-Installation und wird über die opsi-Distributionsrepositories aktualisiert. Das Programm ist code-identisch mit der lokalen Version und wird hierbei als Java-Webstart-Anwendung zur Verfügung gestellt. Der Aufruf erfolgt entweder im Browser über

- den Webstartlink auf der Server-Übersichtsseite <https://<servername>:4447>
- über die URL <https://<servername>:4447/configed>
- oder den direkten Zugriff auf <https://<servername>:4447/configed.jnlp>

Die Webstartanwendung kann auch ohne Browser mittels (z.B. einer Verknüpfung nach)

`javaws https://<servername>:4447/configed.jnlp`

aufgerufen werden.

Sofern es sich um einen Server mit graphischer Oberfläche handelt, ist der *opsi-configed* auch direkt im Server über einen Menüeintrag im Desktopmenü sowie in der Konsole über `/usr/bin/opsi-configed` startbar.

Auf Non-Gui-Servern können lediglich Starts mit den Optionen

- `--version`
- `--help` bzw. `-h`
- `--querysavedsearch [SAVEDSEARCH_NAME]` bzw. `-qs [SAVEDSEARCH_NAME]`
- `--swaudit-pdf FILE_WITH_CLIENT_IDS_EACH_IN_A_LINE [OUTPUT_PATH]`

erfolgen.

Auf jedem System kann, wenn die benötigten jar-Archive verfügbar sind, der Start des *opsi-configed* mittels `java -jar configed.jar` erfolgen.

Insbesondere zeigt dann der Aufruf `java -jar configed.jar --help` die Kommandozeilenoptionen:

```
-l LOC          --locale LOC          Set locale LOC (format: <language>_<country>)
-h HOST        --host HOST           Configuration server HOST to connect to
-u NAME        --user NAME           User for authentication
-p PASSWORD    --password PASSWORD   password for authentication
-c CLIENT      --client CLIENT       CLIENT to preselect
-g CLIENTGROUP --group CLIENTGROUP   CLIENTGROUP to preselect
-t INDEX       --tab INDEX           Start with tab number INDEX, index counting starts with \
0, works only if a CLIENT is preselected
-d PATH        --logdirectory PATH    Directory for the log files
```

```

-r REFRESHMINUTES --refreshminutes REFRESHMINUTES Refresh data every \
  REFRESHMINUTES (where this feature is implemented, 0 = never)
-qs [SAVEDSEARCH_NAME] --querysavedsearch [SAVEDSEARCH_NAME] On command line: tell saved host searches list \
  resp. the search result for [SAVEDSEARCH_NAME])
--gzip [y/n] Activate gzip transmission of data from opsi server yes\
  /no
--sslversion PREFERRED_SSL_VERSION Try to use this SSL/ TLS version
--ssh-key SSHKEY full path with filename from sshkey used for \
  authentication on ssh server
--ssh-passphrase PASSPHRASE passphrase for given sshkey used for authentication on ssh server
--version Tell configed version
--collect_queries_until_no N Collect the first N queries; N = -1 (default, no collect), 0 = \
  infinite
--help Give this help
--loglevel L Set logging level L, L is a number >= 0, <= 5
--halt Use first occurring debug halt point that may be in \
  the code
--sqlgetrows Force use sql statements by getRawData
--nosqlrawdata Avoid getRawData
--localizationfile EXTRA_LOCALIZATION_FILENAME Use \
  EXTRA_LOCALIZATION_FILENAME as localization file, the file name format has to be: configed_LOCALENAME.properties
--localizationstrings Show internal labels together the strings of selected localization
--swaudit-pdf FILE_WITH_CLIENT_IDS_EACH_IN_A_LINE [OUTPUT_PATH] export pdf swaudit reports for given clients (\
  if no OUTPUT_PATH given, use home directory)

```

Läuft der `opsiconfd` auf einem anderem als dem Standardport 4447, so ist dieser beim Start des `opsi-configed` dem Hostnamen mitzugeben (`host:port`).

4.1.1 Logging des `opsi-configed`

Der `opsi-configed` loggt defaultmäßig im Level 3 "Info". Das Loglevel kann auf Level 4 "Check" und Level 5 "Debug" hochgesetzt werden. Zur Änderung des Loglevels gibt es auf die Kommandozeile die Option `--loglevel [LEVEL]`. Die Verwendung von Level 5 ist nur angezeigt, wenn bereits der configed-Start zu Problemen führt, da die produzierte Logdatei sehr umfangreich wird und kaum noch durchgearbeitet werden kann. Wenn der configed läuft und eine mögliche Fehlersituation bei einer bestimmten Anforderung bevorsteht, kann vorher interaktiv über den Menüpunkt Hilfe/ConfigEditor Logstufe das verschärfte Logging eingeschaltet werden.

Im Level 4 werden insbesondere die Service-Calls geloggt. Der Debug-Level bietet häufig die Möglichkeit, Aktionen sehr kleinschrittig zu verfolgen.

Die erzeugten Logdateien liegen per Default im Heimatverzeichnis des Users. Unter Windows werden sie dabei ab Version 4.0.7.6.12 im Verzeichnis

```
c:\Users\[Benutzername]\AppData\Roaming\opsi.org\log
```

abgelegt (wobei im Explorer statt "Users" die lokalisierte Bezeichnung steht, Achtung, der Explorer zeigt standardmäßig das Verzeichnis nur nach manueller Eingabe von AppData an).

Die aktuelle Logdatei heißt `configed.log`, die bis zu drei vorangegangen `configed_0.log`, `configed_1.log`, `configed_2.log`

Per Kommandozeilenparameter `-d` kann das Logging-Verzeichnis umgestellt werden.

Der Pfad zur aktuell genutzten Logdatei wird angezeigt im Hilfemenü unter "Aktueller Logfile". Bei Aufruf des Punktes kann der Pfad in die Zwischenablage übernommen werden, um ihn im Öffnen-Dialog eines Editor zu verwenden; oder es kann direkt die Default-Anwendung für `.log`-Dateien geöffnet werden.

4.1.2 Die Auswahl der Sprache

Der `opsi-configed` versucht für seine Spracheinstellung die Lokalisierungsinformation des Betriebssystems zu verwenden. Sofern er nicht über die passende Übersetzungsdatei verfügt, verwendet er Englisch als Defaultsprache. Ebenso werden, wenn in der Übersetzungsdatei Terme fehlen, die Ausdrücke der englischen Sprachdatei eingesetzt.

Beim Aufruf des `opsi-configed` kann über den Parameter

-l bzw. --locale

eine Sprache im Zweizeichen-Format `language_region` vorgegeben werden, z.B. `en_US` oder `de_DE`. Es genügt aber bereits die Angabe von z.B. `en`, da die Sprachdatei Englisch für jede Variante der Sprache verwendet wird.

Im laufenden *opsi-configed* kann die Sprache über den Menüpunkt Datei/International geändert werden. Dies führt zu einer Reinitialisierung des Programms mit einem Neuaufbau (fast) aller Komponenten in der neuen Sprache.

Schließlich kann mit dem Aufrufparameter

--localizationfile

direkt eine Lokalisierungsdatei vorgegeben werden. Mit dem Zusatzparameter

--localizationstrings

werden auch die Terme angezeigt, die in der Lokalisierungsdatei stehen und übersetzt werden müssen. Dies kann als Hilfe beim Anlegen und Testen einer Übersetzungsdatei genutzt werden.

4.1.3 Benutzerdefinierte Startparameter mit Java Web Start

Die aufgelisteten Startoptionen können auch, mit einer etwas anderen Syntax als auf der Kommandozeile, einem `jnlp`-Aufruf übergeben werden, wie er zum Beispiel in andere Anwendungen eingebettet werden kann.

```
javaws "https://<servername>:4447/configed.jnlp?parameter1=wert1&parameter2=wert2"
```

Die möglichen Parameter entsprechen den Parametern, welche *opsi-configed* auf der Kommandozeile entgegen nimmt - ohne die Striche davor. Zur Vorbelegung des Benutzernamens mit *Roger* kann der folgende Aufruf verwendet werden:

```
javaws "https://<servername>:4447/configed.jnlp?user=Roger"
```

4.2 Login

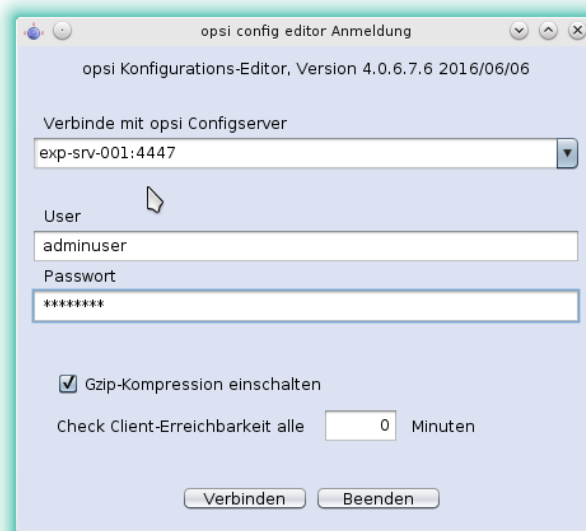


Abbildung 3: Login des *opsi-configed*

Beim Login versucht sich der *opsi-configed* per *https* mit dem *opsiconfd* auf dem ausgewählten Server/Port zu verbinden. Wenn der *opsiconfd* seinen Defaultport 4447 verwendet, muss dieser nicht angegeben werden. Die User müssen der Unix-Gruppe *opsiadmin* angehören.

Der *opsi-configed* speichert im lokalen Benutzerprofil einige Information über die jeweilige Session, damit beim erneuten Login die Arbeitsumgebung wiederhergestellt werden kann, insbesondere eine ausgewählte Client-Gruppe. Seit Version 4.0.7 werden die Session-Informationen auch genutzt, um eine Auswahlliste der zuletzt verbundenen opsi-Server (z.B. produktiver und Test-Server) zu erzeugen. An oberster Stelle steht der zuletzt genutzte, der damit ohne explizite Auswahlaktion wieder verwendet werden kann.

Die gzip-Komprimierung im HTTP-Protokoll verringert die zu übertragende Datenmenge auf Kosten der verlängerten Verarbeitungszeit, da die Daten komprimiert und dekomprimiert werden müssen. Es hat sich gezeigt, dass im lokalen Netz die Reaktionszeiten tendenziell kürzer ohne Komprimierung sind, da die Effekte der verlängerten Verarbeitungszeit überwiegen. Bei Übertragung übers WAN ist es tendenziell umgekehrt. Da im LAN die Kompression sich im Normalfall zeitlich praktisch nicht bemerkbar macht und bei WAN-Verbindungen meist vorteilhaft ist, ist die Gzip-Option per Default eingeschaltet.

Mittels des Features der Client-Erreichbarkeit lässt sich prüfen, welche Clients verbunden sind. Es lässt sich außer über die Anmeldemaske auch per Aufrufparameter aktivieren oder deaktivieren. In der Anmeldemaske ist ein Test-Intervall von 0 Minuten (= ausgeschaltet) als Default eingetragen. Zur Vermeidung einer zu hohen Zahl von Netzanfragen sollte das Test-Intervall nicht zu kurz eingestellt werden.

4.3 Copy & Paste, Drag & Drop

Sie können aus (fast) allen Bereichen des *opsi-configed* die markierten Daten mit den üblichen Tastenkombinationen (*Strg-Insert*, *Strg-C*) in die Zwischenablage kopieren und so anderen Programmen zur Verfügung stellen.

Für die meisten Tabellen ist auch die *Drag & Drop*-Funktion aktiviert, mit der die Tabellendaten z.B. nach Excel transferiert werden können.

4.4 Auswahl des Bedienungsmodus

Um zwischen verschiedenen Funktionen des *opsi-configed* zu wechseln, befinden sich seit Version 4.0.4 rechts oben im *opsi-configed*-Fenster sechs Schaltflächen.



Abbildung 4: opsi-configed: Wahl des Modus

Die durch die ersten drei Schaltflächen wählbaren Modi Clientkonfiguration, Depotkonfiguration (Depot-Properties) oder Serverkonfiguration (Server-Configs) verändern das Zielobjekt des Hauptfensters, die Schaltflächen *Gruppenaktionen*, *Produktaktionen* sowie *Lizenzmanagement* starten jeweils spezifische Fenster mit eigenen Objekten und Aktionsmöglichkeiten.

Diese Fenster können statt über die Schaltflächen auch über den Hauptmenüpunkt *Fenster* geöffnet werden (ab *opsi-configed* Version 4.0.7).

4.5 Depotauswahl

Werden an Ihrem *opsi-server* mehrere Depotserver betrieben, so werden diese in einer Liste am linken Rand des *opsi-configed* angezeigt.

Per default ist das Depot auf dem *opsi-configserver* markiert und die zu diesem Depot gehörigen Clients werden angezeigt. Die Liste der Depotserver ist eine Mehrfachauswahlliste. D.h. mehrere Depots können markiert werden,

es werden alle zugehörigen Clients gezeigt (genauer, alle Clients, die auf irgendeinem der ausgewählten Depots zur ausgewählten Gruppe (s. dort) gehören).

Die gemeinsame Clientanzeige gestattet nur dann, Clients gemeinsam z.B. in ihrer Produktkonfiguration zu bearbeiten, wenn sie Depots mit identischer Paketkonfiguration angehören, d.h. die Depots untereinander synchron sind. Der Versuch, Clients aus asynchronen Depots gemeinsam zu bearbeiten, wird mit einer entsprechenden Warn- und Fehlermeldung abgewiesen.

Für die erleichterte Depotselektion existieren mehrere Funktionen:

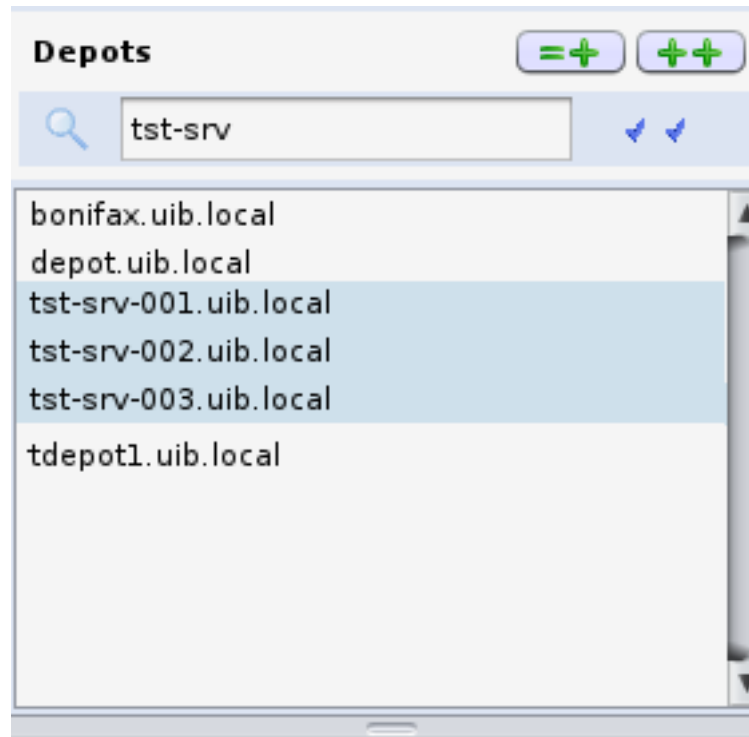


Abbildung 5: *opsi-configed*: Depotauswahl

- (=+) : Alle Depots mit identischen Produkten werden ausgewählt
- (++) : Alle Depots werden ausgewählt (auch mit *Strg-a* möglich)
- Textsuche im Namen der Depots und optional auch in der Beschreibung

Im Suchfeld kann mit dem ersten Checkfeld

Der *opsi-configed* speichert lokal (auf dem PC, auf dem er läuft) und User- sowie Server-bezogen die zuletzt getätigte Depot- und Gruppenauswahl und aktiviert sie beim nächsten Start wieder.

Zu beachten: Beim Depotwechsel bleibt die Gruppenauswahl erhalten, was nicht unbedingt das Gewünschte ist. Gegebenfalls muss daher für das neue Depot eine andere Gruppe bzw. die ganze CLIENT-LISTE aktiviert werden.

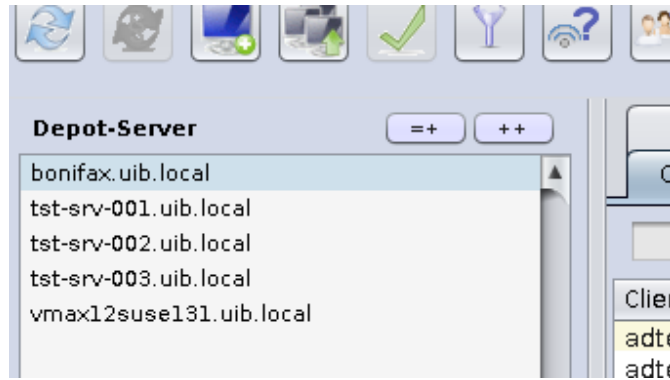


Abbildung 6: opsi-configed: Depotauswahl

4.6 Clientauswahl

Nach erfolgreichem Login zeigt sich das Hauptfenster mit dem aktiviertem Karteireiter *Clients* zur Auswahl von Clients. Das Hauptfenster zeigt die Liste der Clients, wie sie durch die Depotwahl gegeben bzw. im *Treeview* - s. Abschnitt 4.7 - bestimmt ist.

Der *opsi-configed* speichert lokal auf dem Rechner, auf dem er läuft userbezogen die zuletzt getätigte Gruppenauswahl und aktiviert sie beim nächsten Start wieder.

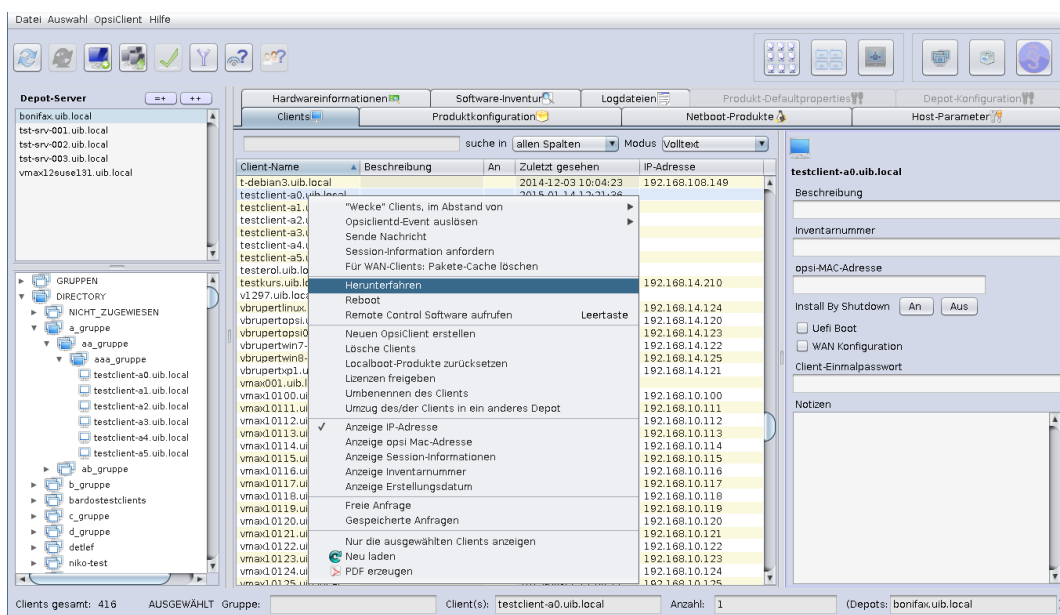


Abbildung 7: opsi-configed: Client-Auswahl

In der Liste kann eine Zeile auch über die Suche nach einem Stringwert ausgewählt werden, der im Suchfeld oberhalb der Liste einzugeben ist.

Wie die Suche ausgeführt wird, ist durch die Auswahl in den Drop-down-Listen zu den Feldern und zum Suchverfahren bestimmt. Die Feld-Auswahl bietet an:

- Suche in allen Feldern (genauer, allen Feldern, die nach der momentanen Konfiguration als Spalten dargestellt werden, (Default) oder
- Suche in einem bestimmten Feld.

Beim Suchverfahren (erweitert in Version 4.0.7) kann man sich entscheiden zwischen den Varianten

- "Volltext": die Sucheingabe wird verwendet wie beim Standard-Goolgen; d.h. wenn die Eingabe mehrere (durch Leerzeichen getrennte) Suchwörter (bzw. Wortbestandteile) enthält, muss mindestens eines in einer Spalte gefunden werden;
- "Volltext (Komplettstring)": die Sucheingabe wird verwendet, als würde man beim Googlen den Text in Anführungszeichen setzen; d.h. der Gesamttext muss irgendwo in einer Spalte gefunden werden;
- "Anfangstext": der Suchstring muss als Beginn einer Spalte vorkommen;
- "regulärer Ausdruck": Suchstring wird sog. regulärer Ausdruck interpretiert, d.h. es werden Zeilen gesucht, bei denen der Text einer Spalte nach den Prinzipien der Regulären Ausdrücke (vgl. die Java-Dokumentation für `java.util.regex.Pattern`) "matcht"

Betätigen der Return-Taste springt auf den nächsten Treffer der Suche (ohne Treffer: nächste Zeile).

Weitere Auswahlfunktionen basierend auf der Suche zeigt das Kontextmenü des Suchfeldes:

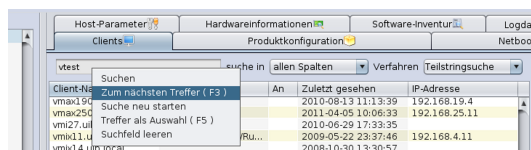


Abbildung 8: opsi-configed: Suchfunktion für Client-Auswahl

Beispiele für Suchmuster (Such-Pattern)

Alle PCs, in deren Name oder Beschreibung die Zeichenfolge *Meyer* vorkommt, werden mit dem Pattern

`.*eyer.*`

gefunden. Dabei steht der Punkt in `.*` für ein "beliebiges Zeichen" und das Sternchen für "eine beliebige Zahl von Vorkommen (des vorher bezeichneten Elements)".

`.*eyer.*`

bedeutet also, das Suchmuster trifft zu (es "matcht"), sofern vor *eyer* irgendetwas (irgendwelche Zeichen in irgendeiner Zahl) steht und auf *eyer* wieder irgendetwas folgt. Weil "irgendeine Zahl" auch 0 sein darf, passt z.B. auch der String *PC Meyer*,

bei dem auf *eyer* gar kein Zeichen mehr folgt, zum Suchstring.

Um aber sicherzugehen, dass wir nicht auch Strings, die *Beyer* enthalten, als korrekt markieren, sollte das Suchmuster besser heißen

`.*[Mm]eyer.*`

Mehrere Zeichen eingeschlossen in eckigen Klammern bedeuten, dass der gesuchte Wert genau eines der aufgeführten Zeichen enthält. Das heißt, jetzt wird jeder String erkannt, der entweder *Meyer* oder *meyer* enthält.

Als weiteres Beispiel sei eine Patternsuche bei Produkten dargestellt.

`0.-opsi.*standard`

sucht alle Produkte, deren Namen mit "0" beginnt gefolgt von *einem beliebigen Zeichen*, gefolgt von *-opsi* gefolgt von *irgendwelchem Zeichen* (in *beliebiger Zahl*); und am Schluss steht *standard*.

Wenn man sichergehen will, dass das zweite Zeichen eine Ziffer, das heißt eines der Zeichen "0", "1", "2", "3", "4", "5", "6", "7", "8", "9" ist, kann man schreiben

`0[0123456789]-opsi.*standard`

Als Abkürzung für `[0123456789]` gilt, da es sich um eine ununterbrochene Teilfolge der Folge aller Zeichen handelt, `[0-9]`, der Ausdruck wird damit zu

`0[0-9]-opsi.*standard`

Zu diesem Suchmuster passen z.B. die Produkte

`03-opsi-abo-standard`

sowie

`05_opsi-linux_standard`

Nähere Information zur Suche mit regulären Ausdrücken können in der java API-Dokumentation, Stichwort "`java.util.regex.Pattern`", gefunden werden.

4.6.1 Die Clientliste

Die tabellarische Auflistung der Clients hat per Default die Spalten *Client-Name*, *Beschreibung*, *An*, *IP-Adresse* und *Zuletzt gesehen*.

- *Client-Name* ist der *full qualified hostname* also der Clientname inklusive (IP-) Domainnamen.
- *Beschreibung* ist eine frei wählbare Beschreibung, die im rechten oberen Teil des Fensters editiert werden kann.
- In der mit *An* betitelten Spalte wird auf Betätigen des Buttons *Prüfen, welche Clients verbunden sind* das Resultat der entsprechenden Anfrage angezeigt. Das Feature lässt sich in der Anmeldemaske sowie per Aufrufparameter aktivieren. Default ist ein Test-Intervall von 0 min (= ausgeschaltet).



Abbildung 9: *opsi-configed*: Client erreichbar



Abbildung 10: *opsi-configed*: Client nicht erreichbar

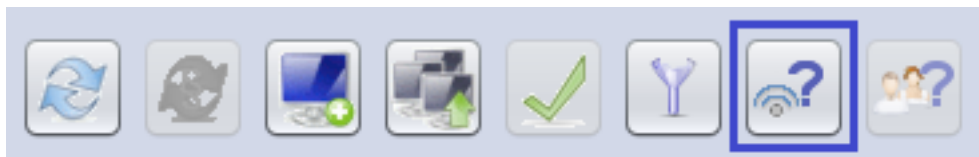


Abbildung 11: *opsi-configed*: Button *Prüfen, welche Clients verbunden sind*

- *IP-Adresse* enthält die IP-Nummer, als die der opsi-Server den Clientnamen auflöst.
- *Zuletzt gesehen* gibt Datum und Uhrzeit an, zu der sich der Client zum letzten Mal bei der Softwareverteilung (über den Webservice) gemeldet hat.

In der Defaultkonfiguration deaktiviert sind die Spalten

- *Session-Informationen* (beziehbar vom Betriebssystem des Clients),
- *opsi-Mac-Adresse* (Hardware-Adresse des Clients, die vom Opsi-Server verwendet wird),
- *Inventarnummer* (optional erfasster kennzeichnender String) und
- *Erstellungsdatum* (Datum und Zeit der Client-Erzeugung).

Während einer Sitzung können sie mittels des Kontextmenüs eingeblendet werden. Welche Spalten beim Start des *opsi-configed* angezeigt werden, kann in der Server-Konfiguration mittels des Configs *configed.host_displayfields* bearbeitet werden.

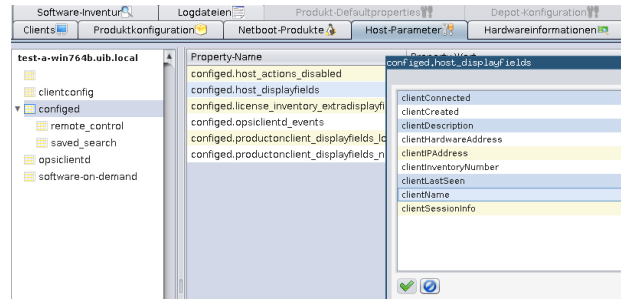
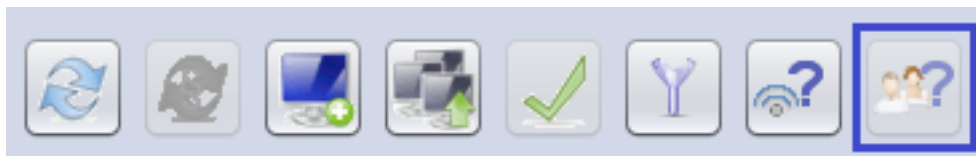


Abbildung 12: opsi-configed: Spaltenkonfiguration für die Clientliste

Das Hinzufügen der Spalte *Session-Informationen* schaltet den Button *Abfrage der Session-Informationen von allen Clients* auf aktiv.

Abbildung 13: opsi-configed: Button *Sessioninfo*

Bei Betätigen des Buttons versucht sich der *opsiconfd* mit allen Clients zu verbinden und Information über die auf den Clients derzeit aktiven (User-) Sessions einzusammeln. In der Spalte *Session-Informationen* wird der Account-Name der laufenden Sitzungen dargestellt. Mittels Kontextmenü sowie über den Hauptmenüpunkt *OpsClient* kann das Entsprechende nur für die gerade ausgewählten Clients erreicht werden. Dies vermeidet ggfs. das Warten auf die Netzwerk-Timeouts beim Verbindungsversuch mit gar nicht angeschalteten Rechnern.

Da die Suchfunktion der Clientliste sich (wenn nichts anderes eingestellt ist) über alle angezeigten Spalten erstreckt, kann nach Abruf der Session-Informationen auch nach dem Rechner gesucht werden, auf dem ein User mit bekanntem Account-Namen gerade angemeldet ist.

Die Clientliste kann durch Anklicken eines Spaltentitels nach der entsprechenden Spalte sortiert werden

4.6.2 Auswahl von Clients

In der Clientliste lassen sich ein oder mehrere Clients markieren und so für die (gemeinsame) Bearbeitung auswählen.

Mittels des Trichter-Icons bzw. über *Auswahl / Nur die ausgewählten Clients anzeigen* kann die Anzeige der Clientliste auf die markierten Clients beschränkt werden

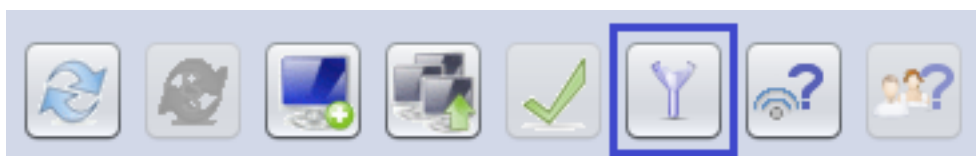


Abbildung 14: opsi-configed: Trichter-Icon

Die ausgewählten Clients können zu einer existierenden Gruppe hinzugefügt werden (die im Treeview sichtbar ist), indem sie mit der Maus "auf die Gruppe gezogen" werden.

Im Client-Auswahldialog, der über *Auswahl / Auswahl definieren* oder über das Icon *Auswahl definieren* gestartet wird, können dynamische Clientzusammenstellungen anhand wählbarer Kriterien erstellt werden.

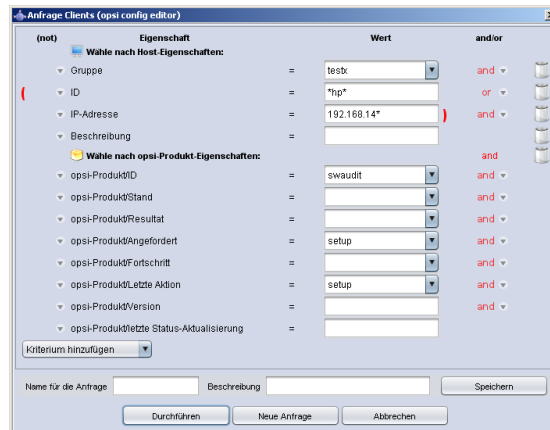


Abbildung 15: opsi-configed: Auswahldialog

Neben allgemeinen Eigenschaften der Clients (Rubrik *Wähle nach Host Eigenschaften*) können als Suchkriterium auch (sowohl auf dem PC gefundene als auch mit Opsi installierte) Software sowie Hardware-Komponenten verwendet werden. Bei Texteingaben kann * als Wildcard benutzt werden. Die einzelnen Kriterien können dabei mit logischem AND (UND) bzw. OR (ODER) verknüpft werden wie auch negiert werden durch ein vorangestelltes NOT (NICHT) (erzeugt durch Anklicken des Not-Feldes vor dem Eigenschaftsnamen)

Die Auswahlliste, betitelt *Kriterium hinzufügen*, stellt weitere Kriterien zur Definition einer Anfrage bereit. Mit dem Papierkorb-Icon am rechten Rand wird ein Suchkriterium entfernt. Den initialen Zustand der Suchmaske stellt ein Klick auf den Button *Neue Anfrage* wieder her.

Die erstellten Anfragen lassen sich unter einem frei wählbaren Namen speichern. Anschließend kann man sie über *Auswahl / Gespeicherte Suchen...* erneut abrufen. Falls beim Speichern das Feld *Beschreibung* ausgefüllt wurde, wird diese in der Auswahlliste als Tooltip angezeigt.

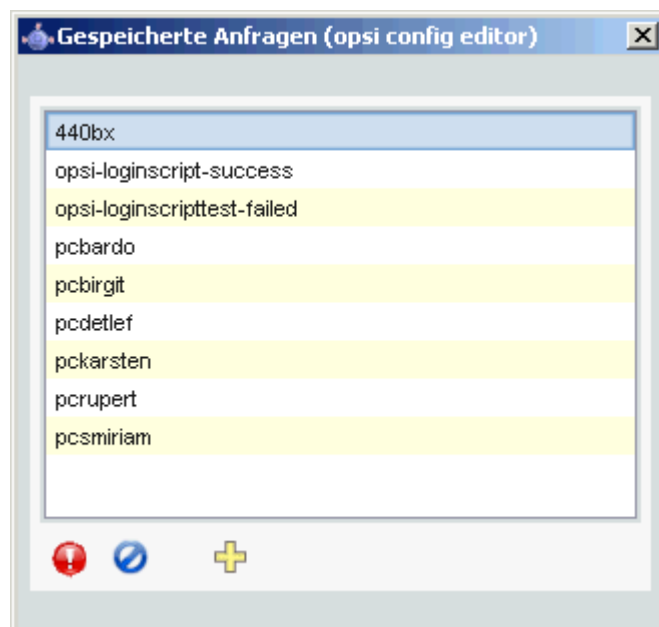


Abbildung 16: opsi-configed: Gespeicherte Suchen

Falls bis zum nächsten Aufruf weitere Clients hinzugekommen sind, die den gespeicherten Suchkriterien entsprechen, werden diese beim nächsten Aufruf der Suche ebenfalls gefunden.

Für eine feste Gruppenbildung mit ganz bestimmten Clients siehe weiter unten Abschnitt 4.7

Zusätzlich ist noch die Abfrage von Suchen über eine Kommandozeilen-Schnittstelle möglich, um sie auch Skripten zugänglich zu machen. Dazu wird der opsi-config-editor mit dem Parameter "-qs" und anschließend dem Namen einer Suche aufgerufen. Lässt man den Namen der Suche weg, wird eine Liste der vorhandenen Suchen ausgegeben.

Unter dem Menüpunkt *Auswahl* die Punkte *Fehlgeschlagene Aktionen bei Produkt...* und *Fehlgeschlagene Aktionen* (heute, seit gestern, ...), gibt es die Möglichkeit, fehlerhafte Installationen zu suchen.

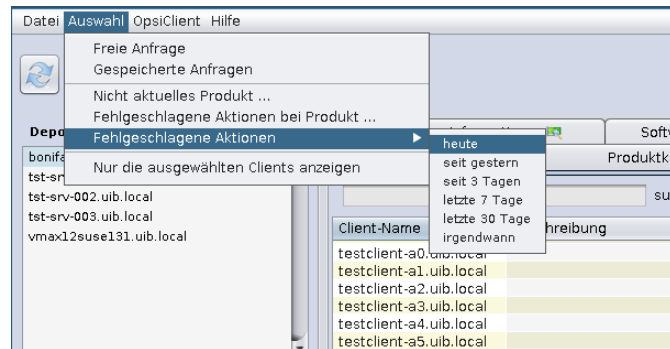


Abbildung 17: opsi-configed: Fehlgeschlagene Aktionen

Wählt man den ersten Punkt, erhält man eine Liste aller Produkte. Selektiert man ein Produkt, so werden alle Clients markiert, bei denen die Installation dieses Produktes fehlgeschlagen ist.

Wählt man z.B. *Fehlgeschlagene Aktionen - heute*, werden all die Clients markiert, bei denen heute mindestens ein Produkt fehlgeschlagen ist.

4.7 Clientauswahl und hierarchische Gruppen im Treeview

Clients können komfortabel gruppenweise verwaltet werden, indem die Baumansicht-Komponente auf der linken Seite des opsi-configed-Fensters genutzt wird.

4.7.1 Prinzipieller Aufbau

Der Treeview hat drei Basisknoten *Gruppen*, *Directory* und *Client-Liste*. In der *Client-Liste* werden automatisch alle Clients der ausgewählten Depots aufgeführt.

Die ersten zwei Basisknoten unterscheiden sich darin, wie oft ein Client in ihnen bzw. in Unterknoten von ihnen vorkommen darf.

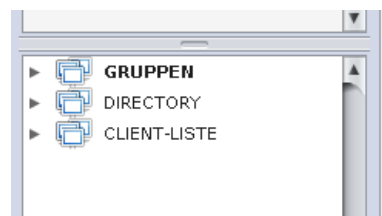


Abbildung 18: opsi-configed: Treeview

Das Prinzip lautet: Während eine Gruppe eindeutig durch ihren Namen bestimmt ist und nicht mehrfach vorkommen kann, kann ein Client beliebig vielen Gruppen angehören. Im *Directory*-Teilbaum hat jedoch jeder Client einen eindeutigen Ort; solange keine explizite Zuweisung zu einer anderen Untergruppe von *Directory* stattgefunden hat, wird der Client automatisch in der Gruppe *NICHT_ZUGEWIESEN* angezeigt.

Wenn ein Client markiert ist, sind alle Gruppen, denen er angehört, mit gefüllter Farbfläche ausgezeichnet.

4.7.2 How to ...

Wenn Sie auf einen Baumknoten (eine Gruppe) klicken, so werden alle Clients, die sich unterhalb dieses Knotens befinden, in der Tabelle im *Client*-Tab angezeigt (aber kein Client zur Bearbeitung ausgewählt).

Wenn Sie auf in der Baumansicht einen Client klicken oder mehrere Clients per Strg-Klick bzw. Shift-Klick markieren, so werden diese im Client-Tab angezeigt und zur Bearbeitung markiert.

Mit *Doppelklick* auf eine Gruppe werden die ihr angehörenden Clients in der Tabelle angezeigt und direkt ausgewählt.

Sie können auf diese Art und Weise auch den Client /oder die in Bearbeitung befindlichen Clients wechseln, während Sie in anderen Tabs (wie z.B. Logdateien) arbeiten, ohne zwischendurch zum Client-Tab zu gehen.

In diesem Treeview werden die Gruppen angezeigt, die Sie gemäß Abschnitt 4.6.2 angelegt haben. Sie können zusätzlich Gruppen definieren, indem Sie mit der rechten Maustaste über der Eltern-Gruppe bzw. dem Eltern-Knoten (z.B. "GRUPPEN") das Kontextmenü öffnen und *Untergruppe erzeugen* wählen.

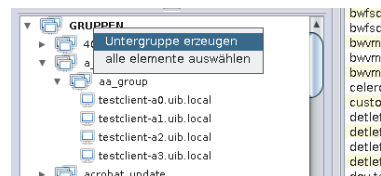


Abbildung 19: *opsi-config*: Gruppe anlegen

Wenn Sie dies gemacht haben, werden Sie in einem Dialogfenster nach dem Namen der Gruppe und einer optionalen Beschreibung gefragt.

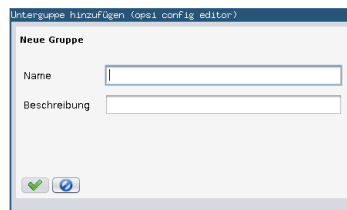


Abbildung 20: *opsi-config*: Gruppenname eingeben

Eine Gruppe können Sie nun mit Clients füllen, indem Sie mit gedrückter linker Maustaste per Drag&Drop:

- Clients aus der tabellarischen Clientliste im Tab "Clients" in die Gruppe ziehen
- Clients aus einer anderen "normalen" Gruppe, z.B. dem Hauptknoten *CLIENT-LISTE*, ziehen = kopieren
- Clients aus dem Treeview aus einer Directory-Gruppe in eine andere Directory-Gruppe ziehen = verschieben

Eine Gruppe kann

- per Drag & Drop in eine andere Gruppe verschoben werden;

Das Kontextmenü einer Gruppe dient dazu

- Untergruppen zu erzeugen;

- die Gruppeneigenschaften zu bearbeiten;
- bei Bedarf die Gruppe (samt ihren Untergruppen und allen Clientzuordnungen) zu löschen;
- alle Clientzuordnungen entfernen, wobei die Gruppe und ihre Untergruppen erhalten bleiben;
- die enthaltenen Clients anzuzeigen und in einem Schritt auszuwählen.

4.8 Client-Bearbeitung

Im Client-Tab können Sie über den Menüpunkt *OpsiClient* oder das Kontextmenü eine Reihe clientspezifischer Operationen starten. Außerdem befinden sich auf der rechten Seite Eigenschaften von dem ausgewählten Client, die bei Auswahl eines Clients editierbar sind.

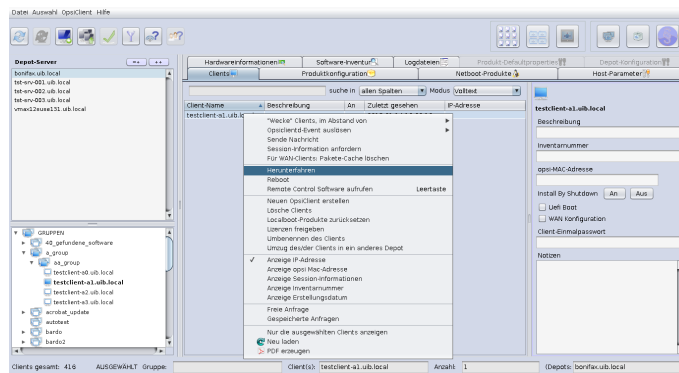


Abbildung 21: *opsi-configed*: Kontextmenü der Clientliste

4.8.1 Install By Shutdown, Uefi-Boot und WAN-Konfiguration

Einige Standard Konfigurationen für einen Client können direkt in der Client-Information gesetzt werden, die sich rechts von der Client-Liste befindet. Zu beachten ist, dass es sich bei der UEFI-Boot-Unterstützung und bei der WAN-Konfiguration um derzeit kostenpflichtige Erweiterungsmodul handelt. Wenn die Module nicht aktiv sind, werden die Buttons ausgegraut.

- **Install By Shutdown:**
Der herkömmliche Weg, Installationen beim Herunterfahren des Rechners zu konfigurieren, ist in Abschnitt 9.17 beschrieben. Dies wird vereinfacht durch die Buttons "An" und "Aus", welche beim Betätigen, den opsi-client-agent mit dem entsprechenden Wert für das Product-Property "on_shutdown_install" auf setup setzen. Bislang ist es nicht möglich den Status anzeigen zu lassen, ob ein Client beim Hoch- oder Herunterfahren die Pakete installiert.
- **Uefi-Boot:**
Die Checkbox Uefi-Boot zeigt an, ob ein Client für den Uefi-Boot konfiguriert ist. Beim Aktivieren wird die Konfiguration `clientconfig.dhcpd.filename` zu `linux/pxelinux.cfg/elilo.efi` geändert. (Weiteres im Abschnitt 9.6)
- **WAN-Konfiguration:**
Der opsi-configed prüft, ob die Standard-WAN-Konfiguration für den Client vorliegt oder nicht. Entsprechend ist der Haken für WAN-Konfiguration gesetzt oder nicht. Und wenn man den Haken manuell setzt, wird die Standard-Konfiguration eingerichtet.

Diese Konfiguration ist seit Version 4.0.7.6.5 nicht mehr hart kodiert, sondern wird aus den Server-Hostparametern `configd.meta_config.wan_mode_off*` ausgelesen bzw. als deren Verneinung interpretiert. Wenn man die automatisch eingerichteten `Meta_Config.wan_mode*`-Parameter beibehalten hat, kommt die in Abschnitt 9.10.5 beschriebene, von der uib GmbH empfohlene Standard-WAN-Konfiguration zum Zuge.

Tip

Ob der Client als WAN-Client konfiguriert ist oder einen UEFI-Boot macht, kann in der Client-Tabelle auch in passenden Spalten dargestellt werden. Für die laufende Session werden sie im Menüpunkt *OpsIClient* oder im Kontextmenü aktiviert, dauerhaft über den Eintrag `configed.host_displayfields` bei den Server-Hostparametern. Dann ist direkt in der Übersicht erkennbar, für welche Clients die Eigenschaft gesetzt ist, und es kann auch danach gesucht und sortiert werden.

Name	Beschreibung	Inventarnr	An	Zuletzt gesehen	WAN co...	UEFI m...
5.uib.local	abc def g			2017-03-21 14:32:29	✓	
1.uib.local	Rupert Roeders Mini-Des...	00:26:18:aa:1f:9b		2017-06-20 14:23:10		
ro01.uib.local	Rupert Röder Desktop			2015-02-10 11:03:28		
rupert08.uib.local				2015-07-08 12:48:59	✓	✓
30-2.uib.local	Lenovo ThinkPad Edge 5...			2016-06-08 12:15:40		
30-3.uib.local	Lenovo ThinkPad Edge 5...			2017-05-08 13:43:54		
ab1.uib.local	test1 fscnoteb1 von imag...	inv 0101		2016-07-15 10:50:55	✓	
ant2.uib.local				2018-01-11 14:39:25		
etb1.uib.local	rupert Samsung Netbook			2014-01-20 13:25:27	✓	✓
-r01.uib.local	Tuxedo Rupert privat			2017-04-11 15:18:47		
rtkurs.uib.local	VM-basierte Testserver wi...			2016-12-19 13:56:46		
rtopsi.uib.local	virtual box testserver auf ...			2014-05-22 09:25:24		
rttom.uib.local	tomcat experiment rupert			2016-01-25 14:44:29		
rtubuntu.uib.local				2017-12-19 14:09:34		
rtuefi.uib.local				2017-01-03 11:56:48		
rtw7.6A.1 uib.local				2016-01-28 17:49:04		

Abbildung 22: *opsi-configed*: Erweiterte Spaltensicht für opsi-Clients

4.8.2 WakeOnLan-Funktionalität mit versetzten Zeiten

Bei der WakeOnLan-Funktion kann ab *opsi-configed* Version 4.0.7 gewählt, werden

- ob das Netzwerksignal direkt an alle ausgewählten Clients geschickt wird
- ob es mit einem Abstand je zwischen zwei Clients versandt werden soll
- wann der Prozess starten soll (Scheduler).

Wenn der Client einem mit dem configserver nicht identischen Depotserver zugeordnet ist, wird das WakeOnLan-Signal nicht direkt an den Client geschickt, sondern es wird eine HTTPS-Verbindung zum *opsiconfd* auf dem Depotserver aufgebaut und dieser veranlasst, in seinem Netz das Netzwerkpaket zu schicken.

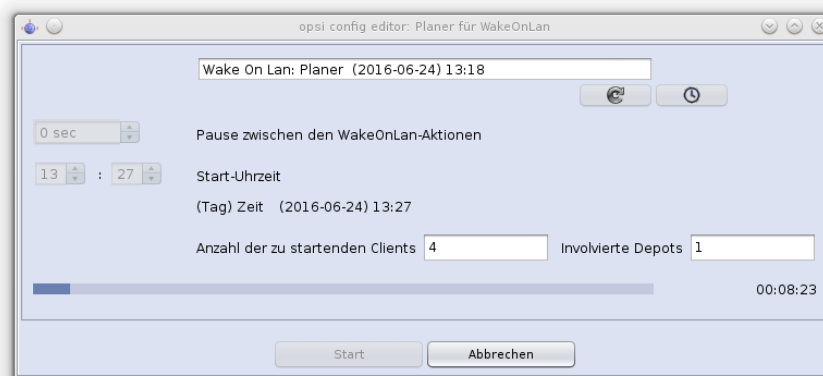


Abbildung 23: *opsi-configed*: Scheduler für Wake On Lan

Zu beachten ist, dass es der *opsi-configed* ist, der die Aktionen auslöst, d.h. das Programm darf in diesem Zeitraum nicht beendet werden.

4.8.3 Auslösen von opsiclientd-Events (Push-Installation)

Über diesen Menüpunkt wird den selektierten Clients ein Aufruf an den *opsi-client-agent* gesendet, durch den ein Event ausgelöst wird. Wenn ein Client nicht erreichbar ist, so meldet der *opsi-configed* dies mit einem Fehlerhinweis. Default-Event ist *on_demand*, das dazu führt, dass Aktionsanforderungen für den Client unmittelbar ausgeführt werden.



Achtung

Wenn ein Produktskript eine Rebootanforderung enthält, so wird der Client ohne Vorwarnung rebootet.

Seit Version 4.0.4 können auch andere Events, die in der *opsiclientd.conf* konfiguriert sind, ausgelöst werden. Welche zur Auswahl angeboten werden, wird über den Server-Hostparameter *configed.opsiclientd_events* festgelegt.

4.8.4 Für WAN-Clients: Pakete-Cache löschen

Auf WAN-Clients kann es zu Problemen mit dem Pakete-Cache kommen. Daher wird für sie eine Funktion angeboten, den Pakete-Cache komplett zu resettet.

4.8.5 *on_demand* Ereignis auslösen (Push Installation)

Über diesen Menüpunkt wird den selektierten Clients ein Aufruf an den *opsi-client-agent* gesendet, dass auf den Clients zu installierende Software jetzt installiert werden soll. Wenn ein Client nicht erreichbar ist, so meldet der *opsi-configed* dies mit einem Fehlerhinweis.



Achtung

Wenn ein Produkt eine Rebootanforderung enthält, so wird der Client ohne Vorwarnung rebootet.

Technisch gesehen wird dem *opsi-client-agent* die Meldung gesendet, er soll das Event *on_demand* auslösen. Das genaue Verhalten ist in der Konfiguration diese Events festgelegt (in der *opsiclientd.conf*).

4.8.6 Nachrichten senden (*Starte Meldungsfenster*)

Über die Auswahl von *Starte Meldungsfenster auf den ausgewählten Clients* erhalten Sie die Möglichkeit, Nachrichten an einen oder mehrere Clients zu senden. Es erscheint zunächst ein Fenster in dem Sie die Nachricht editieren können.

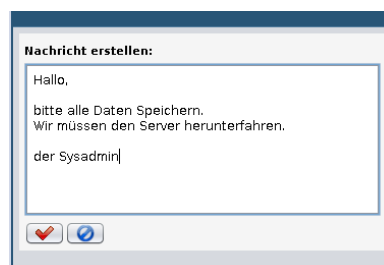
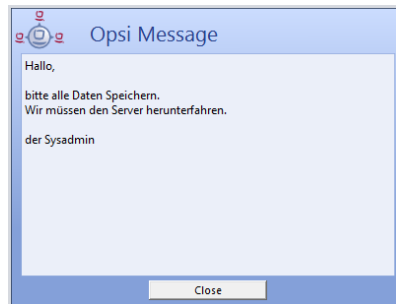


Abbildung 24: *opsi-configed*: Bearbeitungsfenster einer Nachricht

Durch Anklicken des roten Häkchens versenden Sie die Nachricht. Auf den selektierten Clients wird nun die Nachricht angezeigt:

Abbildung 25: *opsi-configed*: Nachrichtenfenster auf dem Client

4.8.7 Sessioninfo der ausgewählten Clients

Sie können den ausgewählten Clients das Signal senden, dem *opsi-configed* ihre Session-Informationen mitzuteilen. Diese Informationen werden in der entsprechenden Spalte dargestellt, soweit diese eingeblendet ist.

4.8.8 Herunterfahren / Reboot der ausgewählten Clients

Sie können den ausgewählten Clients das Signal senden, herunterzufahren bzw. zu rebooten.



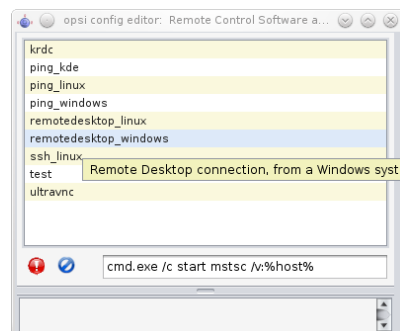
Achtung

Der Client fährt ggf. ohne Rückfrage herunter.

4.8.9 Externe Remotecontrol-Werkzeuge für die ausgewählten Clients aufrufen

Mit der Option *Remote Control Software* im Kontextmenü sowie im Client-Menü (ab *opsi-configed* Version 4.0.1.11) können beliebige Betriebssystem-Kommandos aufgerufen werden, parametrisiert z.B. mit dem Clientnamen.

Beispielhaft sind zwei Konfigurationen zum Anpingen des ausgewählten Hosts automatisch mitgeliefert, und zwar ein Kommando, das unter Windows ausgeführt werden kann, und eines, das in einer graphischen Linux-Umgebung arbeitet. Zur Verdeutlichung: Der *opsi-configed* ruft das jeweilige Kommando aus *seiner* Systemumgebung auf. D.h., die Form des benötigten Kommandos hängt davon ab, ob der *opsi-configed* unter Windows oder Linux läuft.

Abbildung 26: *opsi-configed*: Auswahl des Remote-Control-Aufrufs

Das Auswahlfenster ist dreigeteilt: Oben steht die Liste der Namen der verfügbaren Aufrufe. Es folgt eine Zeile, in der das ausgewählte Kommando angezeigt wird sowie, falls zulässig, verändert werden kann. Die Zeile enthält auch

Buttons für Start und Abbruch der Aktion. Der dritte Textbereich des Fensters gibt eventuelle Rückmeldungen des Betriebssystems beim Aufruf des Kommandos wieder.

Die Möglichkeiten, die sich bieten, sind quasi unerschöpflich. Z.B. kann ein Kommando konfiguriert werden, das eine RemoteDesktop-Verbindung zum ausgewählten Client öffnet (sofern dieser das zulässt). Unter Windows kann dafür z.B. der folgende Befehl verwendet werden:

```
cmd.exe /c start mstsc /v:%host%
```

Ein entsprechendes Linux-Kommando lautet z.B.:

```
rdesktop -a 16 %host%
```

Dabei ist `%host%` eine Variable, die vom *opsi-configed* automatisch durch den entsprechenden Wert für den Hostnamen ersetzt wird. Weitere Variable, die im Kommando verwendet werden können, sind:

- `%ipaddress%`
- `%hardwareaddress%`
- `%opsihostkey%`
- `%inventorynumber%`
- `%depotid%`
- `%configserverid%`

Sofern das Kommando mit *editable true* gekennzeichnet ist, können in der angezeigten Kommandozeile z.B. Passwörter oder andere ad-hoc-Variationen des Befehls eingegeben werden.

**Achtung**

Sobald irgendein Kommando als *editable* gekennzeichnet ist, hat faktisch der configed-Bediener die Möglichkeit, das vorgegebene Kommando in ein *beliebiges* Kommando umzuwandeln und damit beliebige Programme mit Bezug auf den Client-Rechner zu starten.

Sind mehrere Clients markiert, wird das Kommando auf allen ausgewählten parallel ausgeführt.

Die Liste der verfügbaren Kommandos wird über Server-Konfigurationseinträge bearbeitet (vgl. Abschnitt 4.17).

Um ein Kommando des Namens `example` zu definieren, muss minimal ein Eintrag `configed.remote_control.example` (oder `configed.remote_control.example.command`) erzeugt werden. Als Wert des Properties wird das Kommando (ggfs. unter Verwendung von Variablen `%host%`, `%ipaddress%` usw.) gesetzt. Zusätzlich kann ein Eintrag der Form `configed.remote_control.example.description` definiert werden. Der Wert dieses Eintrags wird als Tooltip angezeigt. Mit einem Booleschen Eintrag der Form `configed.remote_control.example.editable` kann durch Setzen des Wertes auf `false` festgelegt werden, dass das Kommando beim Aufruf nicht verändert werden darf.

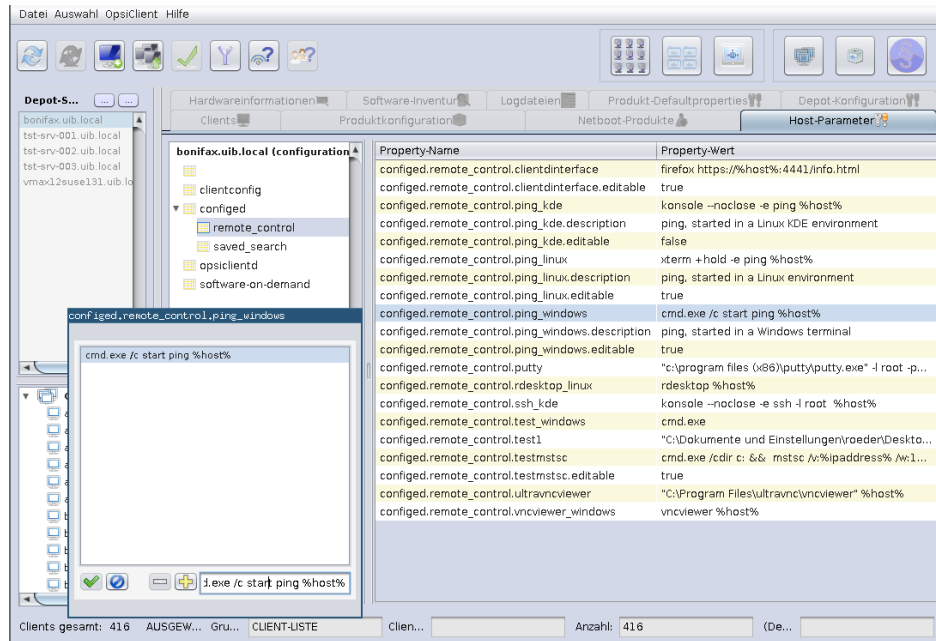


Abbildung 27: *opsi-configed*: Bearbeitung der Remote-Control-Aufrufe bei den Server-Konfigurationseinträgen

4.8.10 Clients entfernen, erstellen, umbenennen, umziehen

Sie können den ausgewählten Clients aus dem opsi-System löschen.

Sie haben hier auch die Möglichkeit, Clients neu anzulegen. Über den Menü-Punkt *Neuen OpsiClient erstellen*, erhalten Sie eine Maske zur Eingabe der nötigen Informationen zur Erstellung eines Clients.

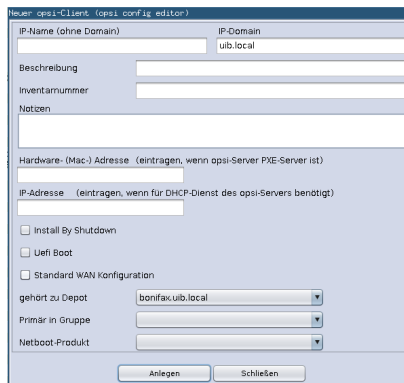


Abbildung 28: *opsi-configed*: Client anlegen

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 Backend gespeichert und die IP-Nummer verworfen.

Beim Anlegen von Clients kann, ab der Version 4.0.5.8.1, für den neuen Client direkt eine Gruppe angegeben werden, zu der er gehören soll, sowie welches Netboot-Produkt eventuell direkt auf setup gesetzt werden soll. Außerdem kann direkt Install by shutdown, Uefi-Boot sowie die (Standard-) WAN-Konfiguration aktiviert werden. Diese Einstellungen können auch ganz einfach in der Hosts-Liste vorgenommen werden.

Seit opsi 4.0.4 können, falls die opsi-Clients z.B. über einen UCS-Service angelegt werden sollen, die Optionen zum Anlegen und Löschen von Clients abgeschaltet werden.

Zur Steuerung dieser Option existiert ein Hostparameter (Config) `configed.host_actions_disabled`, in dem die beiden Listenwerte

- `add client`
- `remove client`

zur Verfügung stehen, wobei Mehrfachselektion möglich ist. Default ist <leer>, also kein Wert ist gewählt.

Sie können die Default-Einstellung im *opsi-configed* über die Serverkonfiguration oder mit folgendem Befehl so ändern, dass ein Erstellen und Löschen von Clients über den *opsi-configed* nicht mehr möglich ist.

```
opsi-admin -d method config_updateObjects '{"defaultValues": ["add client", "remove client"], "editable": false, "\multiValue": true, "possibleValues": ["add client", "remove client"], "type": "UnicodeConfig", "id": "configed.\host_actions_disabled"}'
```

Sie können den ausgewählten Client innerhalb von opsi umbenennen. Sie werden dann in einem Dialogfenster nach dem neuen Namen gefragt.

Ein weiterer Dialog steht für den Umzug von einem Client oder mehreren Clients zu einem anderen Depot zur Verfügung:

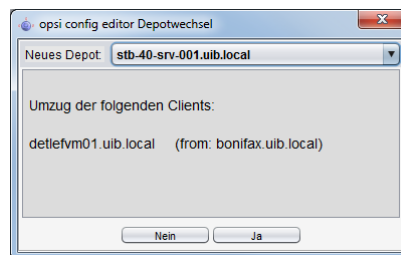


Abbildung 29: *opsi-configed*: Client zu anderem Depot umziehen

4.8.11 Localboot-Produkte zurücksetzen

Sie können alle Informationen zu allen Localbootprodukten der ausgewählten Clients löschen. Dies kann sinnvoll sein um z.B. einen Testclient auf einen definierten Zustand zu setzen.

4.9 Produktkonfiguration

Wechseln Sie auf den Karteireiter *Produktkonfiguration*, so erhalten Sie die Liste der zur Softwareverteilung bereitstehenden Produkte und des Installations- und Aktionsstatus zu den ausgewählten Clients.

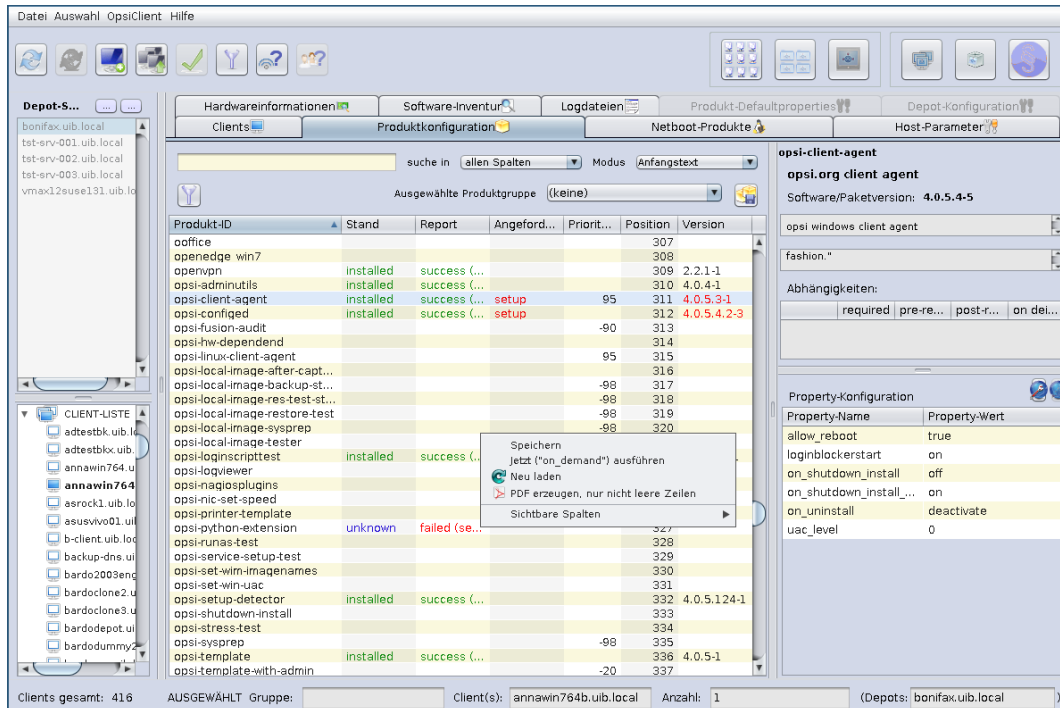


Abbildung 30: opsi-configed: Tab Produktkonfiguration

Seit opsi 4.0.4 wird hier eine Suchfunktion angeboten. Mit der Suchfunktion kann nach Produktnamen bzw. - je nach Einstellung in der Auswahlbox - nach Werten in den Feldern der Produkttabelle gesucht werden (analog zur Suche in der Client-Tabelle). Dazu ist im Suchfeld ein Suchstring einzugeben, die Suche startet direkt und die erste Trefferzeile wird markiert. Wenn es keinen Treffer gibt oder Zeichen aus dem Suchstring gelöscht werden, geht die Markierung auf die erste Tabellenzeile.

Im Kontextmenü des Suchfeldes werden weitere Optionen angeboten.

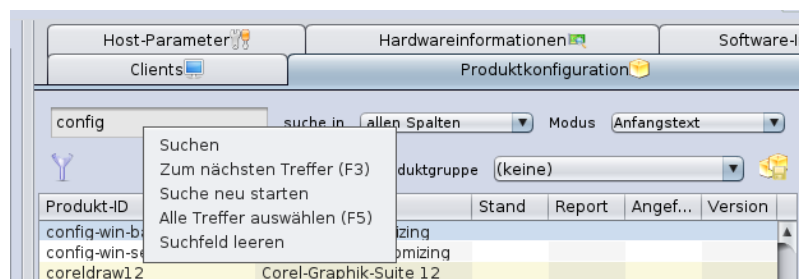


Abbildung 31: opsi-configed: Produktsuche, mit Kontextmenü

Hilfreich kann die Benutzung der Filterfunktion sein. Aktivieren des Filters reduziert die angezeigten Produkte auf die in diesem Moment markierten. Die Auswahl bleibt erhalten, bis der Filter durch erneutes Anklicken des Filterbuttons wieder deaktiviert wird.

Werte in der Produktliste, die für die ausgewählten Clients unterschiedlich sind, werden grau (als Darstellung des Wertes *undefined*) angezeigt.

Sie können auch für die Produktliste eine Sortierung nach einer anderen als der ersten Spalte durch Anklicken des Spaltentitels erreichen. Verfügbar sind die folgenden Spalten:

- *Stand* ist der letzte der Softwareverteilung gemeldete Status zu diesem Produkt und kann die Werte *installed*,

not_installed und *unknown* haben. *not_installed* wird aus Gründen der Übersichtlichkeit nicht angezeigt. *unknown* ist der Status üblicherweise während einer (De-)Installation sowie wenn das letzte Skript gescheitert ist.

- *Report* ist eine Zusammenfassung der Werte der internen Statusinformationen "Installationsfortschritt" (*actionProgress*), "Ergebnis der letzten Aktion" (*actionResult*) und "letzte angeforderte Aktion" (*lastAction*). Während einer Installation steht hier z.B. *installing* Nach Abschluss einer Aktion enthält das Feld den Text *Ergebnis (letzte Aktion)*, z.B. *failed (setup)* oder *success (uninstall)*.
- *Angefordert* ist die Aktion, welche ausgeführt werden soll. Ein möglicher Wert ist immer *none* (visuell ist das Feld leer). Darüber hinaus sind die Aktionen verfügbar, für die bei diesem Produkt Skripte hinterlegt wurden. Möglich sind *setup*, *uninstall*, *update*, *once*, *always*, *custom*.
- *Prioritätsklasse* gibt an, welche Priorität (100 bis -100) dem Produkt zugeordnet wurde (per default ist die Spalte nicht sichtbar.)
- *Position* gibt an, in welcher Reihenfolge die Produkte installiert werden, per Default ebenfalls keine sichtbare Spalte.
- *Version* ist die Kombination aus Produkt-Version und Package-Version, des auf dem Client installierten opsi-Softwareprodukts.

Durch das Anwählen eines Produkts erhalten Sie auf der rechten Seite des Fensters weitere Informationen zu diesem Produkt. Im Einzelnen:

- *ProduktID*:: Klartextname des Produktes
- *Software/Paketversion*: Produktversion-Paketversion der zur Verteilung bereitstehenden Software (wie sie der Paketierer angegeben hat).
- *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 mit Angabe der Art der Abhängigkeit:
 - *required* bedeutet, das ausgewählte Produkt benötigt das hier angezeigte Produkt, es besteht aber keine zwingende Installationsreihenfolge.
 - *pre-required* heißt, das hier angezeigte Produkt muss *vor* dem Ausgewählten installiert werden.
 - *post-required* spezifiziert, das hier angezeigte Produkt muss *nach* dem Ausgewählten installiert werden.
 - *on deinstall* bedeutet, diese Aktion soll bei der Deinstallation des ausgewählten Produktes durchgeführt werden; Reihenfolgen können dabei nicht berücksichtigt werden (weil Installationen und Deinstallationen nicht gemischt vorkommen können, aber ggf. zu entgegengesetzten Reihenfolgeanforderungen führen können).
- *Konfiguration für den Client*: Zur clientspezifischen Anpassung der Installation können für ein Produkt zusätzliche Properties definiert sein. Die Darstellung und Bearbeitung der Tabelle der Properties ist in einem spezifischen Oberflächenelement realisiert:

4.10 Property-Tabellen mit Listen-Editierfenstern

Eine Property-Tabelle ist eine zweispaltige Tabelle. In der linken Spalte stehen Property-Namen, denen jeweils in der rechten Spalte ein Property-Wert zugeordnet ist.

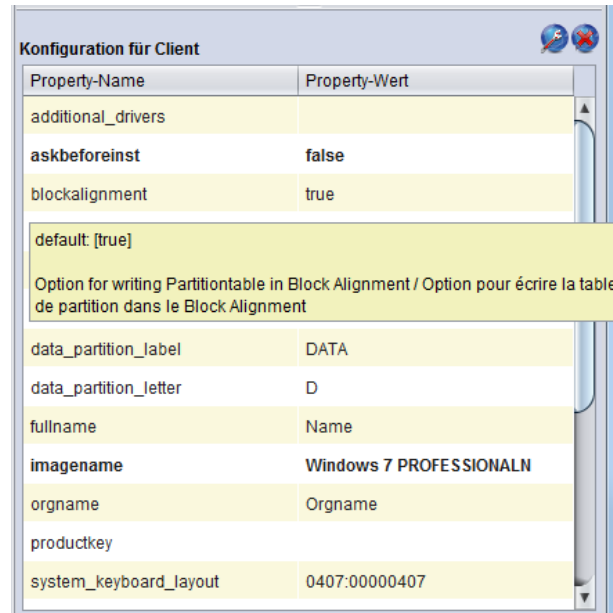
Die Zeilen, die nicht dem Serverdefault entsprechen, werden fett angezeigt.

Rechts oben befinden sich zwei Buttons:

- (rechts): Entferne clientspezifische Werte:
Dieser Button löscht alle Einstellungen beim Client, so dass die Serverdefaults wirken. Werden Serverdefaults geändert wirkt sich dies somit direkt auf den Client aus.

- (links): Setze Clientwerte auf Serverdefaults:
Dieser Button kopiert die Serverdefaults als Clienteinstellungen. Damit bleiben diese Einstellungen beim Client erhalten, auch wenn später die Serverdefaults geändert werden.

Sofern entsprechend konfiguriert, wird ein Hinweis zur Bedeutung eines Wertes sowie der Defaultwert angezeigt, sobald der Mauszeiger über die betreffende Zeile bewegt wird ("Tooltip"):



Property-Name	Property-Wert
additional_drivers	
askbeforeinst	false
blockalignment	true
default: [true]	
Option for writing Partitionable in Block Alignment / Option pour écrire la table de partition dans le Block Alignment	
data_partition_label	DATA
data_partition_letter	D
fullname	Name
imagename	Windows 7 PROFESSIONALN
orgname	Orgname
productkey	
system_keyboard_layout	0407:00000407

Abbildung 32: *opsi-configed*: Property-Tabelle

Beim Anklicken eines Wertes poppt ein Fenster auf, der *Listen-Editor*, und zeigt einen Wert bzw. eine Liste vorkonfigurierter Werten, wobei der derzeit gültige Wert (bzw. die derzeit selektierte Wertekombination) markiert ist:

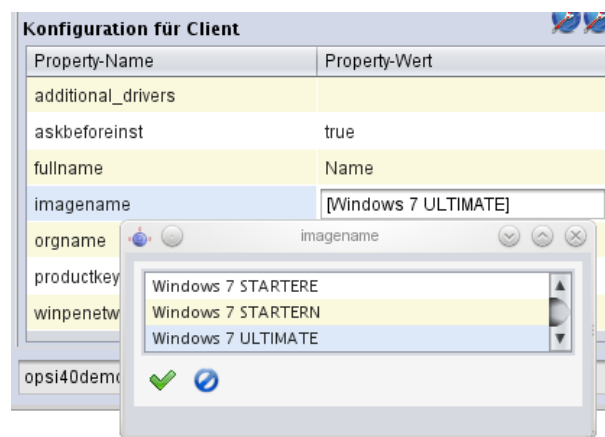


Abbildung 33: *opsi-configed*: Listen-Editor, Auswahlliste

Durch Anklicken eines (anderen) Wertes kann die Auswahl neu gesetzt (bzw. durch Strg-Click erweitert oder verkleinert) werden.

Sofern die Liste der zulässigen Werte erweiterbar ist (die Werte sind somit änderbar), bietet das Fenster zusätzlich ein Editierfeld an, in dem neue bzw. geänderte Werte eingegeben werden können.

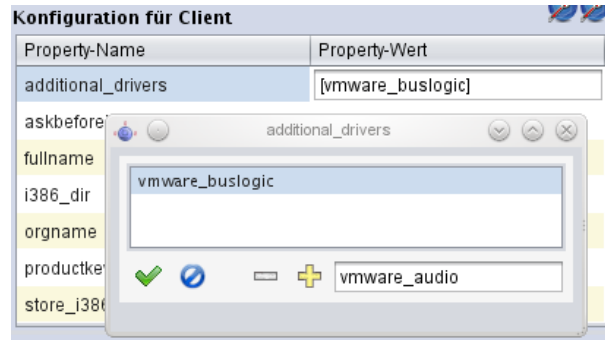


Abbildung 34: *opsi-configed*: Listeneditor, Editierfeld

Wenn ein Wert nur modifiziert werden soll, kann er Wert aus der Liste der vorhandenen mit Doppelklick in das Editierfeld übernommen werden. Sobald ein in der Liste noch nicht vorhandener Wert im Editierfeld steht, aktiviert sich das Plus-Symbol zum Übernehmen des neuen Wertes.

Sofern - wie z.B. beim Property *additional drivers* der Fall sein sollte - Mehrfach-Werte zulässig sind, erlaubt die Liste eine Mehrfach-Selektion. Wenn die Liste als Mehrfach-Selektion konfiguriert ist, wird ein Wert mit Strg-Klick zur Selektion hinzugefügt bzw. ein selektiert entfernt. Mit dem Minus-Button kann die Selektion in einem Rutsch komplett geleert werden.

Wenn die Liste bearbeitet wurde, wechselt wie auch sonst im configed der grüne Haken nach Rot. Anklicken des Hakens übernimmt den neuen Wert (d.h. die neue Selektion als Wert). Der blaue Cancel-Knopf beendet die Bearbeitung, indem der ursprüngliche Wert zurückgesetzt wird.

4.11 Geheime Property-Werte

Für den Fall, dass Passwörter oder andere "Geheimnisse" als Property-Werte vorkommen, ist folgende Vorkehrung getroffen (als "Hack" seit Version 4.0.7, bis ein spezifischer Datentyp eingerichtet ist):

- Wenn im Namen des Properties irgendwo *password* vorkommt
- oder wenn der Property-Name mit *secret* ("geheim" oder "Geheimnis") beginnt,

wird der Property-Wert bei der Anzeige durch fünf * ersetzt. Er wird erst - nach Vorwarnung - sichtbar gemacht, wenn er wie zur Bearbeitung angeklickt wird.

Die Bearbeitung findet ggfs. wie im Standardfall statt.

4.12 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.

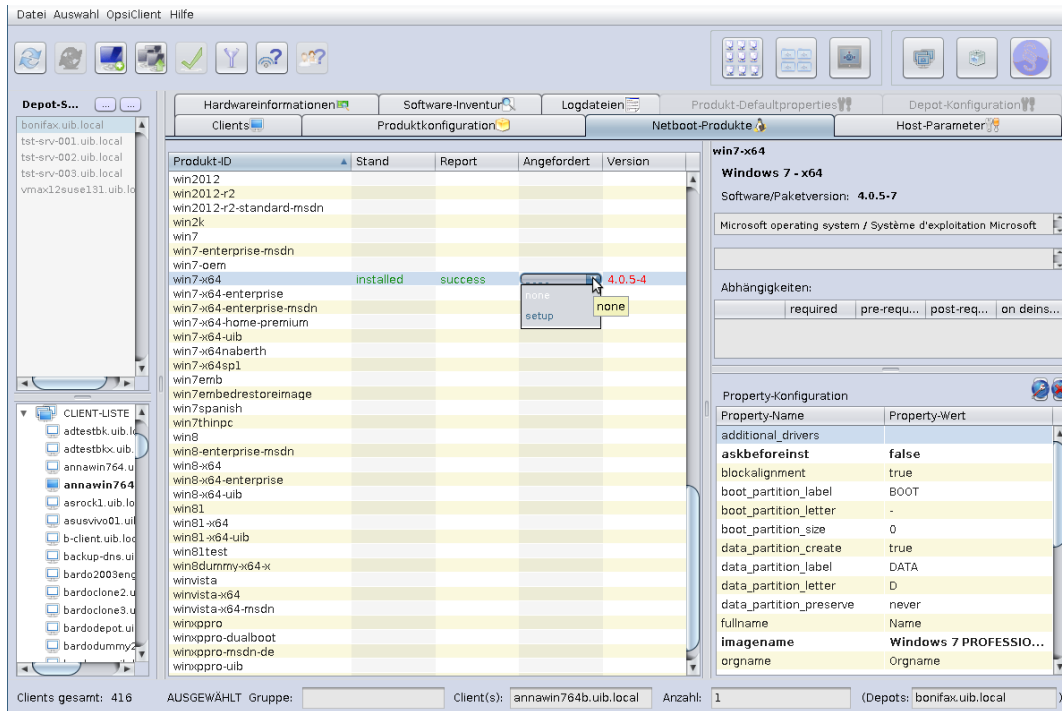


Abbildung 35: opsi-configed: Tab Netboot-Produkte

4.13 Hardwareinformationen

Unter diesem Karteireiter erhalten Sie die letzten - entweder durch das Bootimage oder durch das Localboot-Produkt hwaudit erhobenen - Hardwareinformationen zum ausgewählten Client.

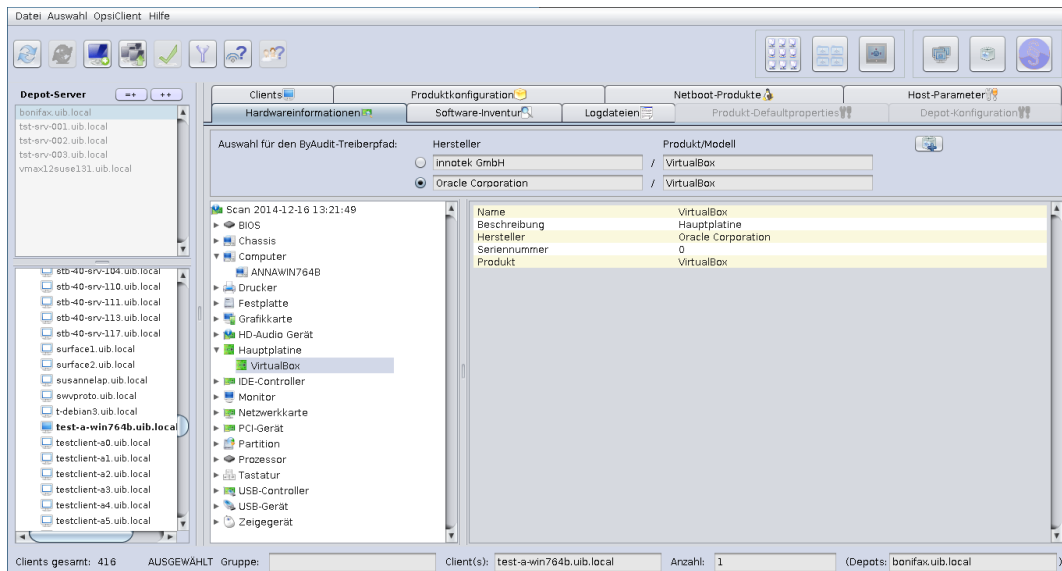
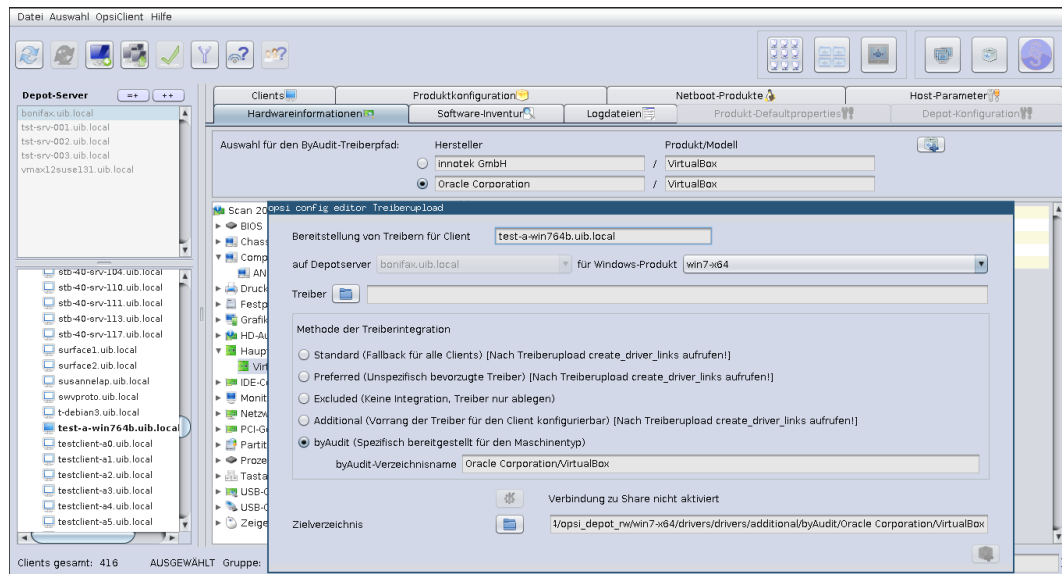


Abbildung 36: opsi-configed: Tab Hardware-Information

4.13.1 Automatisierte Treiberintegration

Um vereinfacht und automatisiert Treiber von speziellen Clients auf den *opsi-depotserver* zu uploaden, gibt es seit der Version 4.0.5 die Möglichkeit aus den Hardwareinformationen mögliche Pfade auszuwählen, die vom opsi-configed über den Share übertragen werden. Die zwei angebotenen byAudit-Treiberpfade setzen sich zusammen aus dem Hersteller und dem Produkt bzw. dem Modell, welche jeweils aus dem Computer und der Hauptplatine ausgelesen werden. Betätigt man den rechten Button zum Treiberupload, so erscheint ein neues Fenster um weitere Einstellungen zu tätigen.



Haben Sie den *opsi-configed* auf einem Linux-System geöffnet, so ist es nicht direkt möglich einen Treiberupload durchzuführen, da die Verbindung über einen Share durchgeführt wird. Dies muss dann manuell getätigt werden. Jedoch sind die Methoden bzw. Verzeichnisstrukturen ein wesentlicher Aspekt der Treiberintegration.

Ohne weiteres funktioniert der Treiberupload von einem Windows-Rechner, wenn die Verbindung zum Share aktiviert ist. Unter anderem müssen im neuen Fenster Angaben getätigt werden, für welches Windows-Produkt der Treiber bereit gestellt werden soll, welche Treiber geuploadet werden sollen und mit welcher Methode bzw. in welches Verzeichnis die Treiberintegration erfolgt. Das Zielverzeichnis wird mit Auswählen einer anderen Methode dem entsprechend geändert. Der zuvor ausgewählte byAudit-Treiberpfad findet sich in der per Default ausgewählten Methode *byAudit* wieder, die den ausgewählten Treiber spezifisch für den Maschinentyp integriert. Folgende Methoden bzw. Verzeichnisse sind möglich:

- *Standard*: Treiber welche im Verzeichnis `./drivers/drivers` liegen, werden anhand der PCI-Kennungen (bzw. USB- oder HD_Audio-Kennung) in der Beschreibungsdatei des Treibers als zur Hardware passend erkannt und in das Windows Setup mit eingebunden. Der Nachteil dieser Pakete ist, das sich hier auch Treiber finden, welche zwar von der Beschreibung zu Ihrer Hardware passen, aber nicht unbedingt mit Ihrer Hardware funktionieren. Hier können Treiber hinterlegt werden, die einen Fallback für alle Clients darstellen.
- *Preferred*: Zusätzliche bzw. geprüfte Treiber gehören in jeweils eigene Verzeichnisse (Name und Tiefe der Verzeichnisstruktur egal) unterhalb des Verzeichnisses `./drivers/drivers/preferred`. Treiber welche im Verzeichnis `./drivers/drivers/preferred` liegen, werden gegenüber den Treibern in `./drivers/drivers` bevorzugt anhand der PCI-Kennungen (bzw. USB- oder HD_Audio-Kennung) in der Beschreibungsdatei des Treibers als zur Hardware passend erkannt und in das Windows Setup mit eingebunden. Finden sich z.B. zu ein und derselben PCI-ID unterschiedliche Treiber unter *preferred*, so kann dies zu Problemen bei der Treiber Zuordnung führen. In diesem Fall ist eine direkte Zuordnung der Treiber zu den Geräten notwendig.
- *Excluded*: Es kann vorkommen, das Treiberverzeichnisse von Herstellern Treiber für unterschiedliche Betriebssystemversionen (Vista / Win7) oder Konfigurationen (SATA / SATA-Raid) enthalten. Wenn Sie die Vermutung haben, das ein verlinkter Treiber falsch ist, so verschieben Sie diesen Treiber in das Verzeichnis `drivers/exclude` und führen `create_driver_links.py` erneut aus. Treiber die in `drivers/exclude` liegen werden bei der Treiberintegration nicht berücksichtigt.

- *Additional*: 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 `./drivers/drivers/additional`. Über das Produkt-Property `additional_drivers` können Sie einen oder mehrere Pfade von Treiberverzeichnissen innerhalb von `./drivers/drivers/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).
- *byAudit*: In dem Verzeichnis `./drivers/drivers/additional/byAudit/<Vendor>/<Model>` hinterlegte Treiber, werden bei einer Windows-Installation nach einem Verzeichnisnamen durchsucht, der dem bei der Hardwareinventarisierung gefundenen *Vendor* entspricht. In diesem *Vendor* Verzeichnis wird nun nach einem Verzeichnisnamen gesucht, das dem bei der Hardwareinventarisierung gefundenen *Model* entspricht. Wird ein solche Verzeichnis gefunden, so wird diese Verzeichnis genauso behandelt, als wären sie über das Produktproperty `additional_drivers` manuell zugewiesen.

Einige Hersteller verwenden Modellbezeichnungen, die für diese Methode sehr ungünstig sind, da man einige Sonderzeichen wie `/` nicht in Datei- oder Verzeichnisnamen verwenden darf. Ein Beispiel dafür wäre als Modelbezeichnung: `"5000/6000/7000"`. Ein Verzeichnis mit dieser Bezeichnung ist wegen den Sonderzeichen nicht gestattet. Seit der dritten Service Release opsi 4.0.3 werden deshalb folgende Sonderzeichen: `< > ? " : | \ / *` intern durch ein `_` ersetzt. Mit dieser Änderung kann man oben genanntes schlechtes Beispiel als: `"5000_6000_7000"` anlegen und das Verzeichnis wird automatisch zu gewiesen, obwohl die Information in der Hardwareinventarisierung nicht der Verzeichnisstruktur entsprechen.

**Wichtig**

Nach dem Treiberupload sollte die `create_driver_links` auf dem *opsi-depotserver* aufgerufen werden!

4.14 Software-Inventur

Unter diesem Karteireiter erhalten Sie die letzten mit `swaudit` ausgelesenen Informationen über installierte Software beim Client.

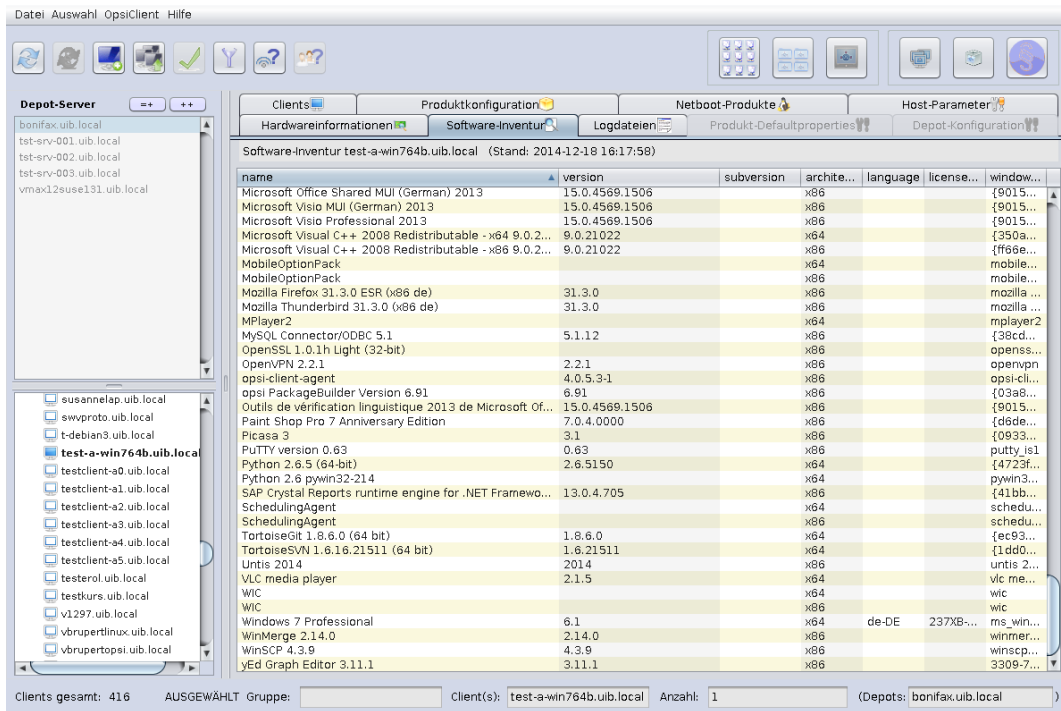


Abbildung 37: opsi-configed: Tab Software-Inventur

4.15 Logdateien: Logs von Client und Server

Im Kartenreiter Logdateien sind die Logdateien der Clients über den opsi-configed einsehbar.

Der Level, bis zu dem die Logeinträge sichtbar sind, kann mit einem Schieberegler variiert werden, so dass Fehler leichter gefunden werden können. Der Regler ist auch mausrad-bedenbar.

Dabei kann auch in den Logdateien gesucht werden (Fortsetzung der Suche mit *F3* oder *Strg-L* = last search repeated). Die einzelnen Zeilen sind je nach Loglevel farbig hervorgehoben.

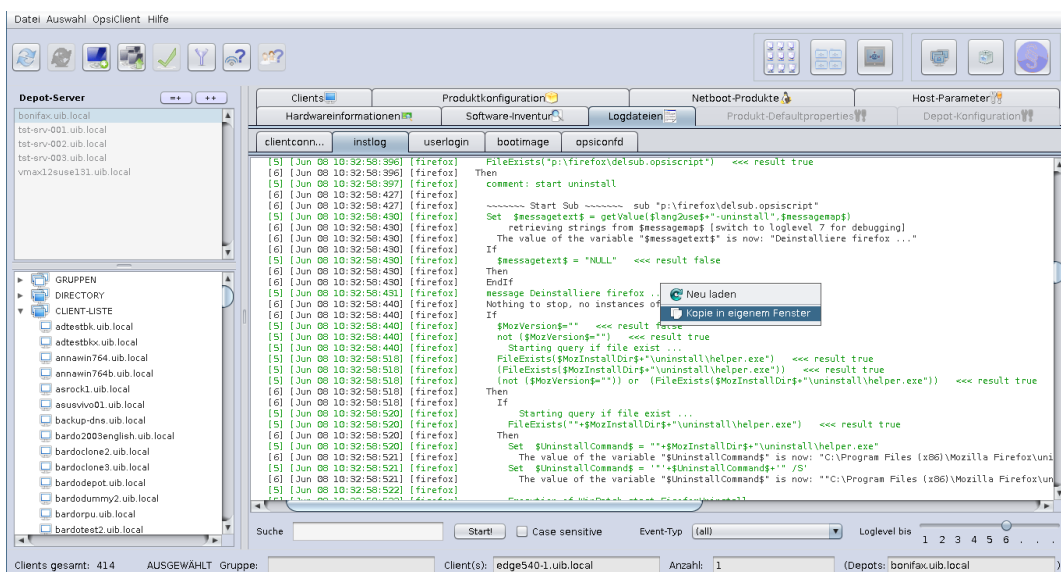


Abbildung 38: opsi-configed: Tab Logdateien

4.16 Produkt-Defaultproperties

Um die Defaultwerte der Produkte für einzelne oder mehrere opsi-depots zu ändern, gibt es einen Reiter *Produkt-Defaultproperties*. Dieser wird erst auswählbar, wenn man zur Depotkonfiguration (zweiter Button oben rechts) wechselt.

In der Haupttabelle werden alle Produkte mit der Produkt-Version und der Package-Version aufgelistet. Wählt man ein Produkt aus, so werden, wie aus der Client-Produktkonfiguration gewohnt (vgl. Abschnitt 4.9), rechts oben die allgemeinen Informationen zum Produkt-Paket angezeigt. Darunter befindet sich die Auflistung aller Depots, die das ausgewählte Produkt besitzen. Die unterhalb befindliche Tabelle mit den Property-Schlüsseln und Werten ist ebenfalls bekannt aus der Client-Produktkonfiguration.

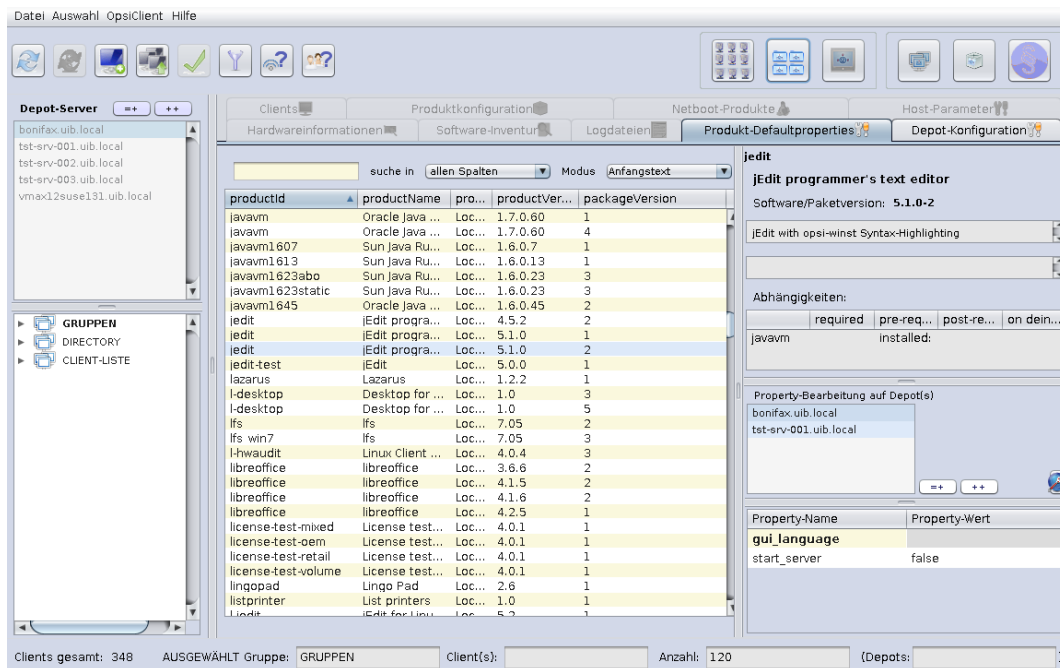


Abbildung 39: opsi-configed: Produkt-Defaultproperties

Man kann ein oder mehrere Depots auswählen, um die Defaultwerte (d.h. die Depotwerte) des Produktes zu ändern. Als Voreinstellung werden alle verfügbaren Depots ausgewählt. Mit den üblichen Tastenkombinationen (*Strg-a*, *Strg-Klick* oder *Shift-Klick*) lassen sich mehrere bzw. alle Clients markieren.

Ist der Property-Wert ausgegraut (siehe Abbildung 39 - „gui_language“), so sind auf den ausgewählten Depots verschiedene Werte für diese Property gesetzt. Rechts neben den Depots befinden sich drei Buttons:

- (=+): Markiere alle Depots mit identischen Werten
Alle Depots, die die gleichen Default-Werte haben, werden markiert.
- (++) : Alle Depots werden markiert.
- (Weltkugel): Setze die Paket-Default-Werte
Die Original-Paket-Werte des Produkts werden für das/die ausgewählte/n Depot/s gesetzt.

4.17 Host-Parameter in der Client- und der Serverkonfiguration

Eine ganze Reihe von Konfigurationsvorgaben können Sie über den Tab *Host-Parameter* setzen und zwar als Server-Defaults im Modus Serverkonfiguration (vgl. Abschnitt 4.4) und clientspezifisch in der Clientkonfiguration.

Konfigurationseinträge (config-Objekte des opsi-Servers) sind grundsätzlich Wertelisten. Als Werkzeug zur Bearbeitung der Werte dient daher der Listeneditor (vgl. Abschnitt 4.10)

Je nach Type des jeweiligen Konfigurationsobjekts

- können die Elemente der Liste entweder Unicode-Textwerte oder boolesche Werte (`true/false`) sein;
- kann die Gesamtheit der zulässigen Listenelemente fest oder erweiterbar sein;
- umfasst der DefaultValues-Eintrag des Objekts im `singleValue`-Fall genau ein Listenelement, bei `multiValue`-Objekten eine beliebige Auswahl aus den zulässigen Listenelementen.

Neue Konfigurationseinträge können über das Kontextmenü auf der Server-Hostparameter-Seite erstellt werden. Ebenso können dort bestehende Objete gelöscht werden.

Das Verhältnis von Server- und Client-Hosteinträgen ist verwickelt:

- Server-Einträge liefern die Default-Werte für die Client-Einträge.
- Konsequenterweise muss, um einen Client-Eintrag zu erhalten, zuerst ein Server-Konfigurationsobjekt angelegt werden.
- Und, wenn ein Server-Eintrag (das Config-Objekt) gelöscht wird, verschwinden auch die zugehörigen Client-Einträge (die auf `ConfigState`-Objekten beruhen).
- Es kann sowohl bei Abweichung des Client-Wertes vom Server-Default als auch, wenn die beiden derzeit identisch sind, einen eigenen Eintrag des Clients in der Datenbank geben. Sofern es diesen eigenen Eintrag gibt, bleibt er auch erhalten, wenn sich der Server-Default ändert.
- Zum Umgang mit diesem Thema gibt es daher ab `configed` Version 4.0.7.6.5 für die Client-Properties ein Kontextmenü mit den beiden Optionen "Spezifischen Client-Wert entfernen" (d.h., der Clientwert ist immer der gerade aktuelle Default-Werte des Servers) und "Setze den jetzigen Default-Wert als (dauerhaften) Wert für den Client."
- Wenn der Client-Wert sich vom aktuellen Server-Default unterscheidet, so ist dieser fett dargestellt.
- Es kann Konfigurationsobjekte geben, für die theoretisch Client-Werte erzeugt und bearbeitet werden können, die jedoch keinerlei Bedeutung haben, weil eigentlich nur serverbezogene Informationen gespeichert werden. Diese Properties sind in den aktuellen `configed`-Versionen in der Regel ausgeblendet.

Zur besseren Übersicht sind die Host-Parameter gegliedert nach Funktionsgruppen aufgeführt. Die Gruppen werden auf der linken Seite in einer baumartigen Struktur aufgeführt. Jeweils die zugehörigen Parameter und ihre Werte erscheinen auf der rechten Seite.

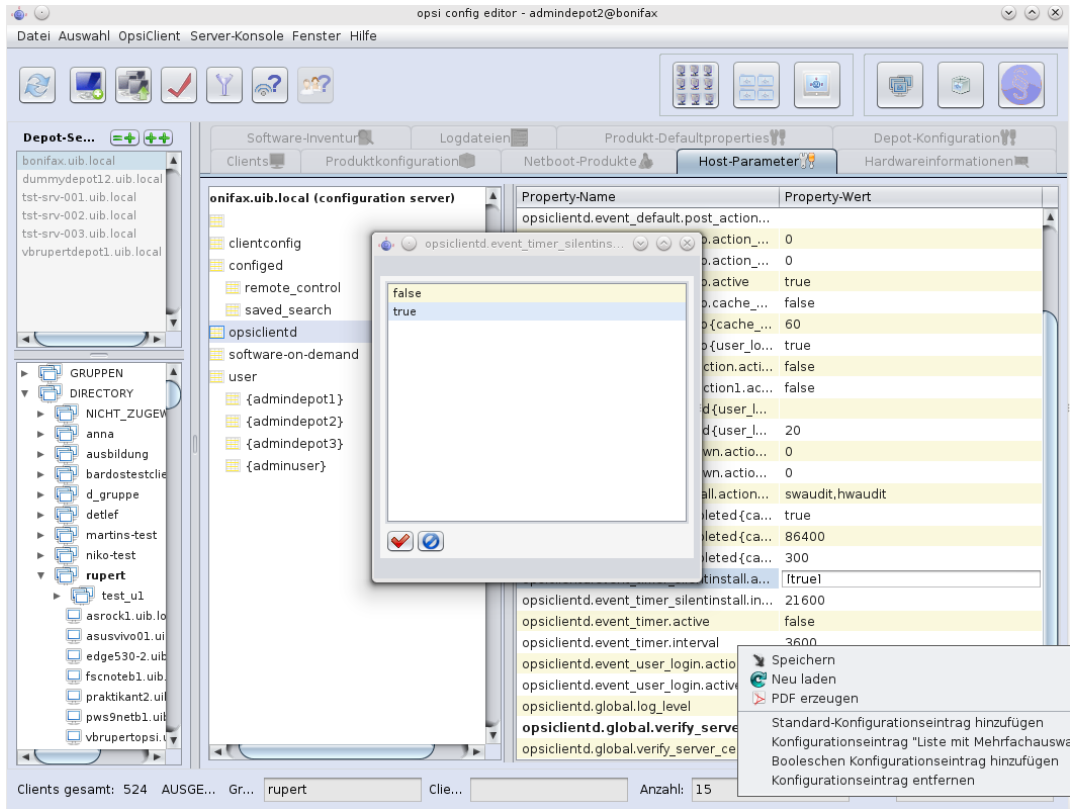


Abbildung 40: opsi-configed: Tab Hostparameter (als Serverkonfiguration)

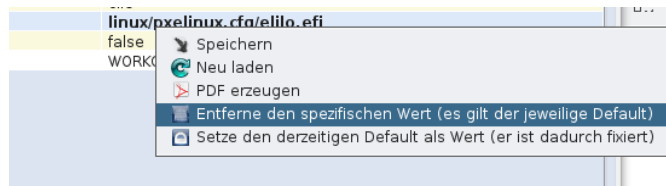


Abbildung 41: opsi-configed: Tab Hostparameter (Kontextmenü eines Clienteintrags)

4.17.1 Verwaltung von User-Rechten und -Rollen

Ab Version 4.0.7.5 bietet der *opsi-configed* die Userrollen-Funktion an.



Achtung

Zur Verwendung des Features muss *user roles* in der *modules*-Datei freigeschaltet sein.

In der Oberfläche zeigt die Existenz der Kategorie *user* in der Übersicht der Server-Hostparameter, dass diese Funktion verfügbar (aber noch nicht unbedingt aktiv) ist. Im Baum der Eigenschaften steht direkt in *user* primär der boolesche Eintrag

```
user.{}.register
```

mit dem Defaultwert *false*.

Die anderen Einträge, die an dieser Stelle des Property-Baums stehen, sind die Defaultwerte für die auch userspezifisch mögliche Konfiguration der Server-Konsole (vgl. Abschnitt 4.21):

Zur Aktivierung der Userrollen-Funktionalität ist

1. der Wert von *user.{}.register* auf *true* zu setzen;
2. eine Modules-Datei einzuspielen, die das Feature *userroles* temporär oder dauerhaft aktiviert hat.

Bei aktiver Userrollen-Funktionalität wird für den angemeldeten User ein Eintrag im Eigenschaftsbaum erzeugt. Die dabei für die Rechteverwaltung verwendeten Defaulteinstellungen entsprechend den "klassischen" Vorgaben für einen Administrator, d.h. zunächst werden dem User keine Einschränkungen auferlegt. Z.B. für einen User *admindepot1* werden die Einträge

<code>user.{admindepot1}.privilege.host.all.registered_readonly</code>	<code>[false]</code>
<code>user.{admindepot1}.privilege.host.depotaccess.configured</code>	<code>[false]</code>
<code>user.{admindepot1}.privilege.host.depotaccess.depots</code>	<code>[]</code>
<code>user.{admindepot1}.privilege.host.opsiserver.write</code>	<code>[true]</code>

generiert.

Die vier Einträge bedeuten:

- *admindepot1* hat *nicht* lediglich readonly-Zugriff auf den Server (ein reiner readonly-Zugriff wäre möglicherweise für einen Mitarbeiter aus dem Helpdesk-Bereich angemessen);
- Depotrestriktionen existieren nicht bzw. werden nicht berücksichtigt;
- entsprechend kann die Liste der für den User zugänglichen Depots leer bleiben (aber auch wenn hier eine Auswahl aus den verfügbaren eingetragen ist, hat dies, solange *depotaccess.configured false*, keine Wirkung);
- der User darf configserver-Einstellungen aller Art bearbeiten.

Soll künftig *admindepot1* nur noch Zugriff auf die Rechner im Depotserver *depot1* haben, ist folgendes einzustellen:

- *host.depotaccess.configured* ist auf *true* zu setzen;
- in die Liste *host.depotaccess.depots* ist der Wert "depot1" einzutragen.

Nach einem (vollständigen) Datenreload sind für den User *admindepot1* keine Clients aus anderen Depots mehr sichtbar (und auch nur noch Depoteinstellungen für *depot1* zugänglich).



Achtung

admindepot1 kann alle Restriktionen selbst aufheben, solange er über das Privileg *host.opsiserver.write* verfügt.

Um die Abriegelung komplett zu machen, ist daher für *admindepot1* noch

- *host.opsiserver.write* auf *false* zu stellen.



Achtung

Die auf diese Weise gesetzten Privilegien beschränken ausschließlich die Funktionalität des *opsi-configed*. Bei anderen Zugriffsmethoden auf das JSON-RPC-Interface des opsi-servers werden sie derzeit nicht wirksam.

4.18 Depotkonfiguration

Im Modus Depotkonfiguration (vgl. Abschnitt 4.4) öffnet sich der Tab *Depots*. Nach Auswahl eines Depotservers in der Drop-Down-Liste können Werte bearbeitet werden, die das Depot parametrisieren.

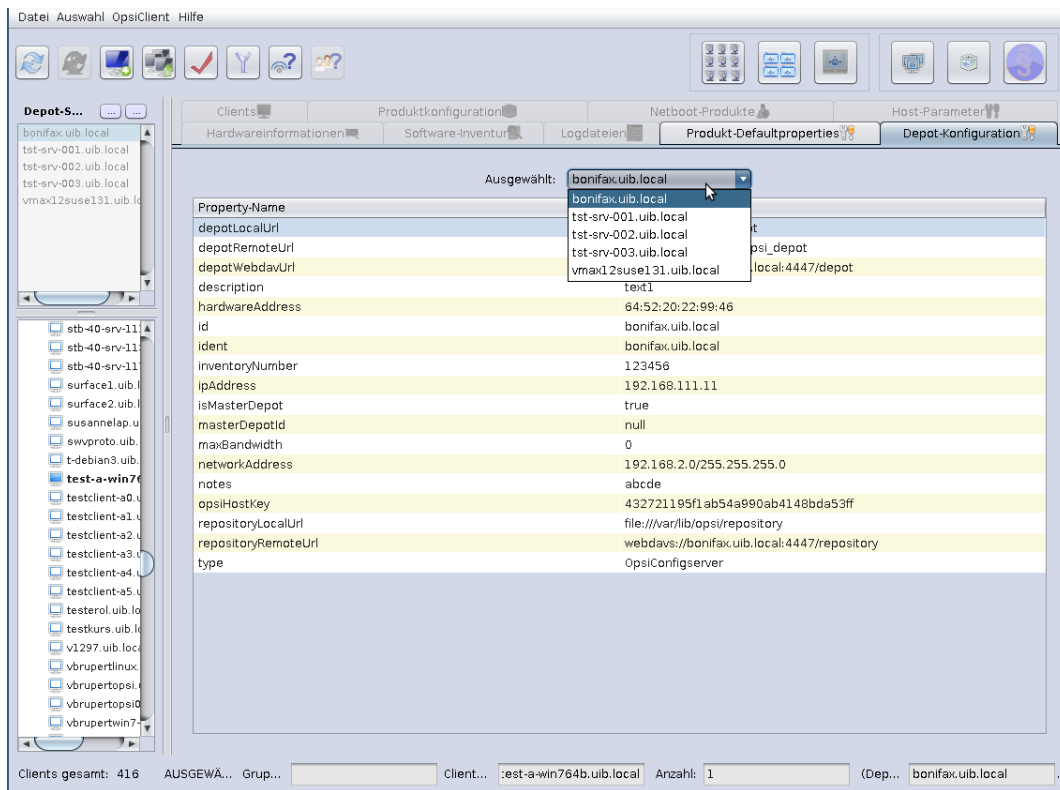


Abbildung 42: *opsi-configed*: Tab Depot-Konfiguration

4.19 Gruppenaktionen

Mittels der Schaltfläche "Gruppenaktionen" (vgl. Abschnitt 4.4) wird ein entsprechendes Fenster für gruppenbezogene Funktionen geöffnet.

Im Moment wird nur eine Funktion angeboten, die für die opsi-localimage-Option interessant ist:

- die Suche nach einem Betriebssystem, das auf allen PCs der selektierten Gruppe schon installiert war und daher zur Auswahl für die Inbetriebnahme (auf allen PCs der Gruppe) angeboten werden kann.

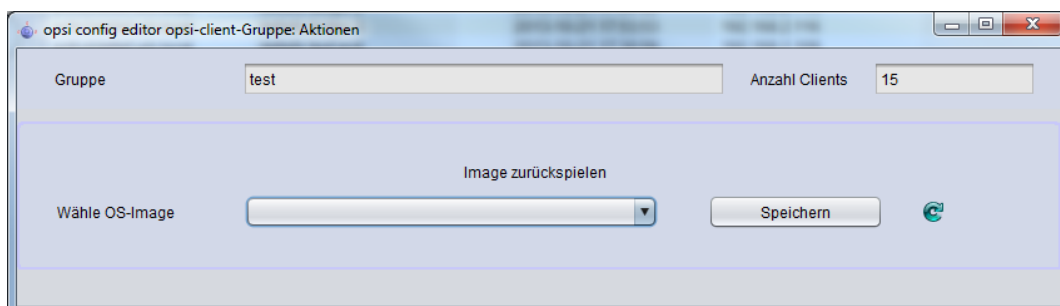


Abbildung 43: *opsi-configed*: Gruppenaktionen (für opsi-local-image)

4.20 Produktaktionen

Mittels der Schaltfläche "Produktaktionen" (vgl. Abschnitt 4.4) wird ein entsprechendes Fenster für Produkt-/Paket-bezogene Aktionen geöffnet.

Im Moment stehen zwei Funktionen zur Verfügung:

- Eine .opsi-Datei (ein opsi-Paket) kann ausgewählt werden bzw. der Pfad zu ihr eingegeben werden. Das Paket kann auf den opsi-Server hochgeladen werden; der standardmäßig vorhandene Netzwerk- (Samba-) Share opsi_workbench auf dem opsi-Server ist dabei der Default-Wert für das Zielverzeichnis auf dem Server. Schließlich wird bei Betätigen des Buttons versucht, das Paket mit einem Aufruf analog einem opsi-package-manager-Aufruf auf dem Server als Produkt zu installieren.
- Die WinPE-Dateien bzw. die Installfiles für ein Windows-Produkt (ab Windows Vista) können in das Server-Produktverzeichnis (Share opsi_depot) hochgeladen werden, so dass die Windows-Produkte nicht mehr serverseitig angefasst werden müssen.

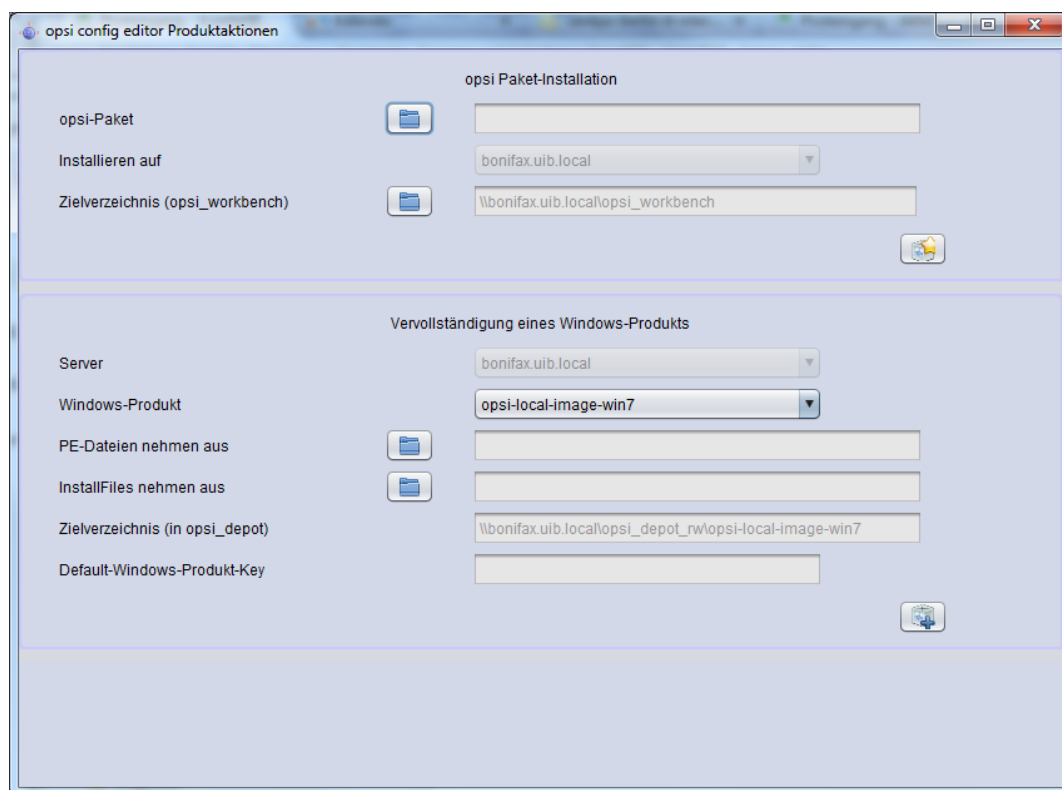


Abbildung 44: *opsi-configed*: Paket- bzw. Produktaktionen

4.21 Server-Konsole



Achtung

Einige der nachfolgend beschriebenen Configed-Funktionen stehen erst ab der python-opsi-Version 4.0.7.38 zur Verfügung. Dazu zählt vor allem die Befehlsbearbeitung (Abschnitt 4.21.4) sowie die Ausführung der darüber hinterlegten Befehle im configed.

Mit der Version 4.0.7.5 ist der configed um einen neuen Hauptmenüeintrag erweitert, die "Server-Konsole". Damit wird die Möglichkeit geboten, vom configed aus auch über eine SSH-Verbindung Aktionen auf dem opsi-server aus-

zulösen. Zum einen kann ein Terminal auf dem opsi-server gestartet werden, zum anderen existieren Menüpunkte für vordefinierte Kommandozeilenbefehle.

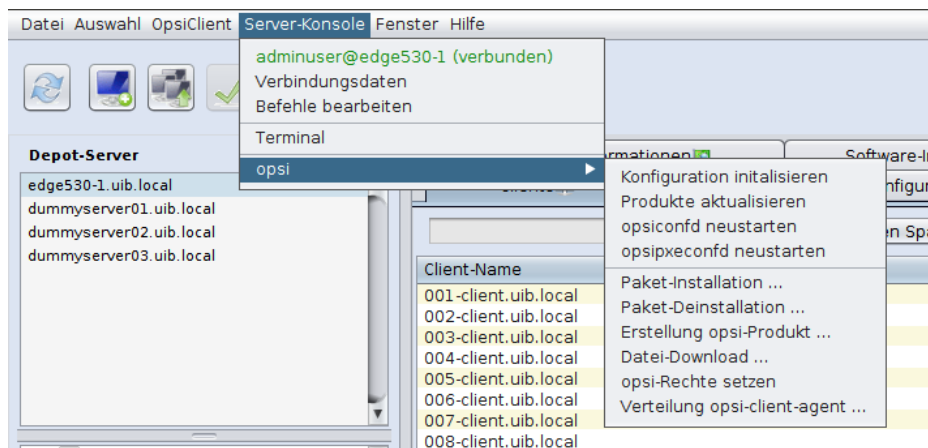


Abbildung 45: opsi-configed: Menü: Server-Konsole

4.21.1 Verbindungsdaten und Berechtigungen

Wenn nicht anderes konfiguriert ist, wird versucht, für die SSH-Verbindung den im configed angemeldeten User und den verbundenen Configserver zu verwenden.

Ist dies nicht erwünscht, kann die Verbindung über einen SSH-Schlüssel (ggf. mit Passphrase) beim Start des Configeds gestartet werden. Zu diesem Zweck existieren die folgenden Aufrufparameter für den configed:

- --ssh-key PATH: z.B. --ssh-key /home/user/.ssh/id_rsa
- --ssh-passphrase PASSPHRASE: z.B. --ssh-passphrase meinpassword

Die Einstellungen lassen sich unter dem Menüpunkt "Verbindungsdaten" ändern.

Über Server-Hostparameter (User-Abschnitt) kann feingranular gesteuert werden, welche der Menüpunkte sichtbar sind bzw. genutzt werden können. Bei aktivierter Userroles-Funktionalität (vgl. Abschnitt 4.17.1) werden die Konfigurationen userspezifisch verwendet. Für einen neu angelegten User werden dabei als Default-Werte zunächst die Werte von der allgemeinen User-Ebene gesetzt.

Folgende Configs existieren:

- user.{}.ssh.serverconfiguration.active:
Aktiviert das Menü der SSH-Verbindungseinstellungen . (Default: false)
- user.{}.ssh.commandmanagement.active:
Aktiviert die Bearbeitung von Menüeinträgen der SSH-konsolen-Befehle (Default: false)
- user.{}.ssh.menu_serverconsole.active:
Gibt den Hauptmenüeintrag "Server-Konsole" als solchen frei. (Default: true)
- user.{}.ssh.terminal.active:
Schaltet die SSH-Shell frei. (Default: true)
- user.{}.ssh.commands.active:
Schaltet alle SSH-Menüeinträge frei, die hinterlegte Befehle darstellen. (Default: true)

4.21.2 SSH-Terminal

Mit Hilfe des Terminals können Linux-Systembefehle auf dem verbundenen SSH-Server ausgeführt werden.

Das Eingabe-Echo kann durch Sternchen (*) ersetzt werden (Passwort-Eingabe).

Ein gestarteter Prozess lässt sich mit dem Button "Prozess/Verbindung beenden" oder durch "Strg+C" abbrechen.

Wie von der Kommandozeile gewohnt, kann die "TAB"-Taste Befehle vervollständigen. Achtung: Pfade werden nicht vervollständigt - lediglich Linux-Systembefehle.

Außerdem können Datenquellen aus dem configed-Kontext angegeben werden, die bei der Befehlsausführung durch die konkreten Werte ersetzt werden. (Mehr zu dieser Funktionalität: Abschnitt 4.21.4 - Punkt: Datenquellen)

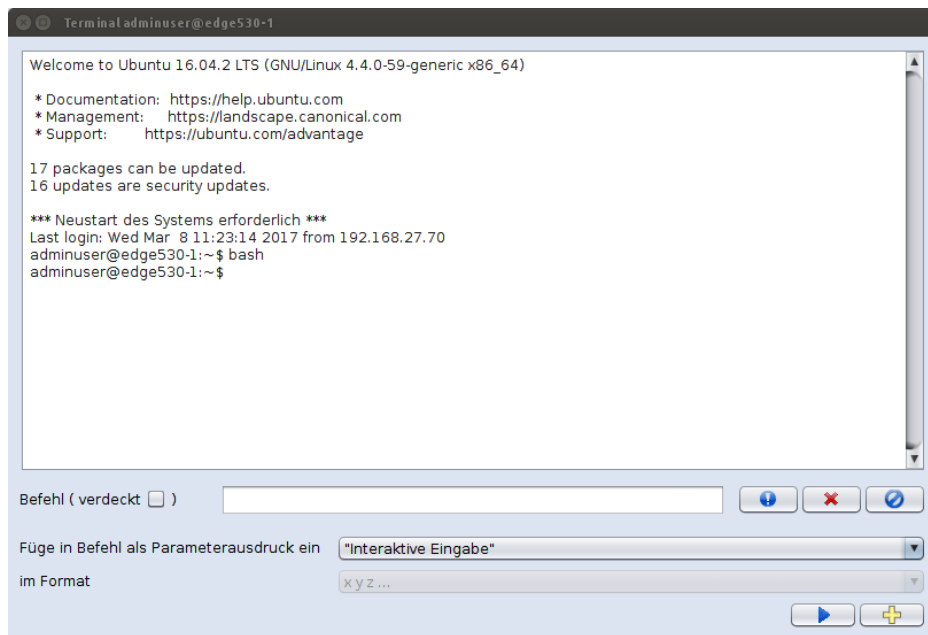


Abbildung 46: *opsi-configed*: SSH-Terminal

4.21.3 Vordefinierte Befehle mit Eingabemasken

Unter der Menügruppe "opsi" stehen einige Befehle unabhängig von den selbst definierbaren Befehlen mit einer eigenen Eingabe-Oberfläche zur Verfügung. Diese vereinfachen den Umgang mit diversen Skripten.

- Datei-Download . . .
Eine beliebige Datei kann aus dem Internet mittels "wget"-Befehl heruntergeladen und in ein Zielverzeichnis auf dem Server abgelegt werden. Anwendung findet der Befehl beispielsweise für konkrete opsi-Pakete von download.uib.de.
- Erstellung opsi-Produkt . . .
Voraussetzung für diesen Befehl ist ein opsi-utils-Paket mit der Version $\geq 4.0.7.7$. Über diesen Menüpunkt kann dann, mit Angabe eines Server-Verzeichnis, aus diesem ein opsi-Paket erstellt werden. Außerdem können die in der Control-Datei gefundenen Versionen (Paket- und Produkt-Version) über einen Button angezeigt und überschrieben sowie eine md5-Summe und/oder eine zsync-Datei erstellt werden. Beim Ausführen wird eine .opsi-Datei gebaut. Zwecks Installation auf dem Server bzw. Depot leitet der Button "Das neu erstellte Paket installieren" zum opsi-package-manager weiter.
- Opsi-Rechte setzen . . .
Dieser Menüpunkt bildet den opsi-Befehl opsi-set-rights ab. Nach der Möglichkeit einen bestimmten (optionalen) Pfad einzugeben in dem das Skript ausgeführt werden soll, wird nach dem root-Passwort gefragt und das Skript im separaten Fenster ausgeführt.

- **Paket-Installation ...**
Durch diesen Befehl können opsi-Pakete mittels "opsi-package-manager" auf allen Depots oder einem Depot installiert werden. Dabei kann der Server-Pfad zum Paket angegeben werden, in dem sich das opsi-Paket befindet.
Durch Auswahl eines Pakets aus dem Internet wird die Funktionalität des Befehls "Datei download ..." aufgegriffen und anschließend das heruntergeladene Paket auf dem Depot installiert. Außerdem sind die Parameter "--update" und "--setup" des opsi-package-manager implementiert. Sollen die zsync und md5-Dateien eines opsi-Paketes heruntergeladen werden, kann der Schalter "zsync- und md5 einschließen" aktiviert werden. Dann wird die url der Pakete entsprechend ergänzt und die zusätzlichen Dateien werden ebenfalls gezogen.
Mehr zum opsi-package-manager finden Sie unter Abschnitt [5.3.2](#)
- **Paket-Deinstallation ...**
Aus einer Liste der installierten Pakete kann eines ausgewählt und deinstalliert werden.
Siehe Abschnitt [5.3.2](#)
- **Verteilung opsi-client-agent ...**
Möchte man existierende Rechner zu opsi hinzufügen, muss der opsi-client-agent auf dem Zielrechner installiert werden. Wählt man die betreffenden Clients im configed aus und ruft diesen Befehl auf, werden die Client-Namen in das entsprechende Feld übernommen. Wenn der Befehl für mehrere Clients mit einem Aufruf ausgeführt werden soll, müssen die Login-Daten bei den beteiligten Rechnern gleich sein.
Achtung: Das Skript zum Deployen der Client muss im Verzeichnis `/var/lib/opsi/depot/opsi-client-agent/` liegen und den Namen `opsi-deploy-client-agent` haben. Genauere Informationen finden Sie im Handbuch *opsi-getting-started* im Kapitel *Erste Schritte*.

Tipp

Einige Oberflächen beinhalten eine Auswahlkomponente für Pfade in der Verzeichnisstruktur. Wird der Button "Ermittle Unterverzeichnisse" betätigt werden alle Verzeichnisse bzw. Dateien aufgelistet, die in dem angegebenen Pfad enthalten sind. Für weitere Ebenen muss der Button wiederholt betätigt werden. Diese Funktionalität befindet sich u.a. in der "Opsi-Rechte setzen" oder "Paket-Installation"-Oberfläche.

4.21.4 Befehle definieren

Zusätzlich zu den vordefinierten Befehlen lassen sich eigene Serverkonsolenbefehle erstellen bzw. verändern oder entfernen, die über Menüpunkte aufgerufen werden können. Dabei ist zu beachten, dass unterschiedliche Linux-Systeme unter Umständen nicht die gleichen Kommandozeilenbefehle ausführen können. Deswegen muss der Administrator sicher stellen, dass die Befehle entsprechend dem adressierten Linux-System funktionieren.

Abbildung 47: *opsi-configed*: Befehle definieren

Folgende Daten müssen bzw. können (gekennzeichnet durch ein "*") für einen Befehl angegeben werden:

- Menü-Text:
Beim Erstellen eines Befehls *muss* darauf geachtet werden, dass der Menütext noch nicht für einen anderen Befehl verwendet wurde. Wenn ein Menütext geändert werden soll, muss der Befehl mit dem Minus-Button gelöscht und ein neuer Befehl erstellt werden.
- Beschreibung*:
Wird eine genauere Beschreibung hinterlegt, taucht diese als Tooltip-Text des Befehls auf.
- übergeordnetes Menü*:
Legt fest, in welchem Menü der neue Befehl als Menüeintrag erscheinen soll. Bleibt das Feld leer, wird der Menüeintrag direkt dem Menü "Server-Konsole" zugeordnet.
- Position*:
Die Position bestimmt die Reihenfolge (kleine Zahlen zuerst) der Menüpunkte insgesamt und damit innerhalb des jeweiligen Menüs. Ist eine alphabetische Reihenfolge erwünscht, müssen alle Positionen identisch gesetzt sein (z.B. alle 0). Bleibt das Feld leer, wird standardmäßig die Position 0 vergeben.
- "sudo"-Rechte*:
Wenn einer der Befehle in der Befehlsliste administrative Rechte benötigt, muss das Häkchen bei "Benötigt root-Rechte" gesetzt sein, dann werden die Befehle in der Liste automatisch um das Schlüsselwort "sudo" ergänzt.
- Befehlsliste:
Bei der Befehlsliste müssen die Linux-Befehle zeilenweise eingetragen werden, damit sie sequentiell ausgeführt werden können. Achtung: Der Befehl kann mittels Button direkt auf dem verbundenen SSH-Server getestet / ausgeführt werden, ohne ihn als Menüpunkt zu erstellen.
- Datenquellen* (für die Befehlsliste):
Zusätzlich können Methoden als Datenquelle hinterlegt werden, d.h. vor Ausführung des Befehls werden Parameter mit dem Ergebnis einer Methode überschrieben. folgende Methoden sind möglich:

- Interaktive Eingabe:
Es ist möglich, für die Befehle Parameter nicht fest vorzugeben, sondern für eine interaktive Eingabe zu kennzeichnen. Dies geschieht in der Form "<<<Dieser Text wird dem Benutzer angezeigt und durch die Benutzereingabe ersetzt>>>" , wobei empfohlen wird, eine Beispielseingabe für den Parameter zum Benutzertext hinzuzuschreiben.
 - Ausgewählte Client-Namen / Ausgewählte Client-IP-Adressen
 - Ausgewählte Depot-Namen / Ausgewählte Depot-IP-Adressen
 - Configserver-Name
 - Verbundener SSH-Servername
- Hinweis: Außer für die "Interaktive Eingabe" kann die Rückgabe der Methoden formatiert werden, beispielsweise in eine kommaseparierte Liste. In der Oberfläche kann die Datenquelle sowohl getestet, als auch an die im Feld der Befehlsliste markierten Stelle eingefügt werden.

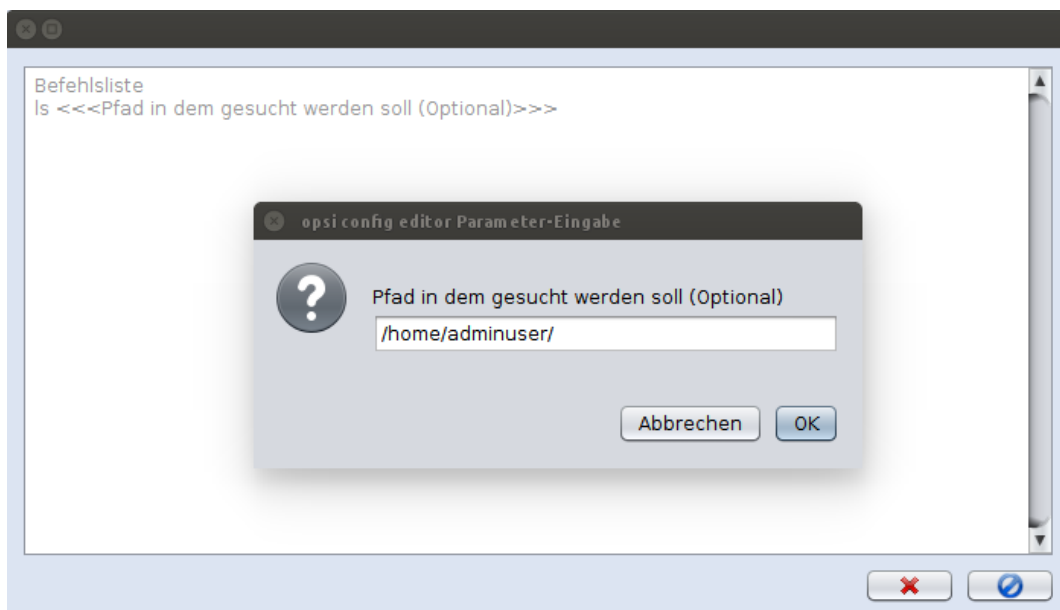


Abbildung 48: *opsi-configed*: Befehl ausführen - Parameterabfrage

Tipp

- Unter Linux lassen sich Befehle mit Hilfe von zwei kaufmännischen Unds ("&&") kombinieren. Dabei muss jedoch darauf geachtet werden, dass zum zweiten Befehl, falls benötigt ein sudo ergänzt wird, da dies nicht automatisch geschieht. Beispiel: Benötigt root-Rechte:"aktiviert" , Befehlsliste: "apt-get update --yes && sudo apt-get upgrade --yes".
- Während der Ausführung lassen sich keine Benutzereingaben tätigen. Es ist notwendig, alle Eingaben über die Befehls\parameter zu steuern (Beispiel: "--yes" -Option bei "apt-get upgrade")

5 opsi-server

5.1 Überblick

Die Funktionalitäten eines *opsi-servers* lassen sich auf vielen Business gängigen Linuxdistributionen installieren: Grob lassen sich zwei wichtige Funktionalitäten unterscheiden, die auf einem Server vereint sein können:

- *opsi-configserver*
Die Funktionalität des configserver umfasst die Speicherung und Verarbeitung der Konfigurationsdaten in unterschiedlichen Backends und deren Bereitstellung über einen Webservice und auf Kommandozeile.

- *opsi-depotserver*

Die Funktionalität des depotserver umfasst die Speicherung der eigentlichen Installationsdateien der zu verteilenden Software, Betriebssysteme, Bootimages und deren Bereitstellung für den Client per smb, cifs/https, tftp.

Die aus diesen Diensten entstehenden Hardwareanforderungen sind in der Regel gering, so das ein Betrieb eines opsi-servers in einer Virtualisierungsumgebung kein Problem darstellt.

5.1.1 Installation und Inbetriebnahme

Die Installation und Inbetriebnahme eines *opsi-servers* ist in dem gesonderten Handbuch: *opsi-getting-started* ausführlich erläutert.

5.1.2 Samba Konfiguration

Um den Client-PCs Zugriff auf die 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 eingesetzt. Die Korrekte Sambakonfiguration können Sie erstellen bzw. reparieren durch den Aufruf von:

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

Nach einer Änderung der Samba-Konfigurationsdateien ist ein reload der Samba-Software notwendig (`service samba reload`).

5.1.3 Der Daemon opiconfd

Der *opiconfd* ist der zentrale Konfigurations-Daemon von opsi. Alle Client-Komponenten (*opsi-client-agent*, *opsi-configed*, *opsi-linux-bootimage*, ...) verbinden sich mit diesem Service um auf die Konfigurationen in den Backends zuzugreifen. Der *opiconfd* wird über die Datei `/etc/opsi/opiconfd.conf` konfiguriert. Die einzelnen Konfigurations-Optionen sind in dieser Datei dokumentiert. An dieser Stelle finden sich weitere Ergänzungen.

- `[global] admin networks:`

Über diese Option kann der administrative Zugriff auf den *opiconfd* auf Verbindungen von bestimmten Netzwerkadressen eingeschränkt werden.

Es können mehrere Netzwerkadressen durch Kommas getrennt angegeben werden.

Nicht-administrative Client-Verbindungen können auch aus anderen Netzwerken erfolgen.

- `[global] max log size:`

Über diese Option kann die Größe der Logdateien beschränkt werden.

Aus historischen Gründen ist dies auf 5MB pro Logdatei beschränkt.

Seit opsi 4.0.6 ist es möglich diesen Wert an eigene Bedürfnisse anzupassen.

Um die Größenbeschränkung zu deaktivieren, kann der Wert auf 0 gesetzt werden.

Ergänzend zu dieser Einstellmöglichkeit ist es möglich durch das Werkzeug *logrotate* die Logdateien auf einem Server automatisch zu komprimieren und rotieren zu lassen.

Bitte entnehmen Sie dem zugehörigen Handbuch die Konfigurationsmöglichkeiten.

Ist die maximale Größe der Logdateien bekannt, lässt sich berechnen wieviel Speicherplatz die anfallenden Logs benötigen werden. Es gibt fünf verschiedene, Client-bezogene Log-Arten die vom *opiconfd* geschrieben werden: *bootimage*, *clientconnect*, *instlog*, *opiconfd* und *userlogin*. Es gibt außerdem einige Client-unabhängige Logs: *opiconfd.log*, *opsipxeconfd.log*, *opsi-backup.log*, *opsi-product-updater.log* und *package.log*.

Wenn wir von einer mit *opiconfd* und *logrotate* realisierten Konfiguration ausgehen, bei der alle Logdateien auf 5MB limitiert sind, abgesehen von *package.log*, welche 10MB groß werden darf, dann kommen wir auf folgende Berechnung:

$(\text{Anzahl der clients} * 5 * 5\text{MB}) + 5\text{MB} + 5\text{MB} + 5\text{MB} + 5\text{MB} + 10\text{MB}$

Bei 100 Clients sollten wir bis zu 2530MB für Logs von opsi reservieren. Weil Logrotate üblicherweise zu einer bestimmten Uhrzeit aktiv wird, empfehlen wir diese Zahl aufzurunden.

5.1.4 Notwendige System-User und Gruppen

- User *opsiconfd*
Dies ist der user unter dem der opsiconfd Deamon läuft.
- User *pcpatch*
Dies ist der user, den der *opsi-client-agent* verwendet um den *depotshare* zu mounten und von diesem zu lesen. Dieser User hat per Voreinstellung das Heimatverzeichnis */var/lib/opsi*. Setzen und Ändern Sie das Passwort mit `opsi-admin -d task setPcpatchPassword`.
- 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.
- 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.

5.1.5 Notwendige Shares

- Bereich: *Depotshare* mit Softwarepaketen (*opsi_depot*)
Auf dem depot-Share liegen die für die Installation durch das Programm opsi Winst vorbereiteten Softwarepakete. In der Voreinstellung liegt dieses Verzeichnis auf dem opsi-server unter */var/lib/opsi/depot*. Unterhalb von diesem Verzeichnis findet sich das Verzeichnis *install* und in diesem für jedes Softwarepaket ein Verzeichnis mit dem Namen des Softwarepakets. Wiederum unterhalb dieses Verzeichnisses liegen dann die Installationskripte und -dateien. Das Verzeichnis wird mit dem Freigabe-Namen *opsi_depot* per Samba read-only exportiert.

Anmerkung

In alten Versionen von opsi war das entsprechende Verzeichnis */opt/pcbin* und der Share hieß *opt_pcbin*.

- 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.



Achtung

Unter Distributionen der SUSE-Familie ist dieses Verzeichnis unter */var/lib/opsi/workbench* zu finden.

- Bereich: Konfigurationsdateien File-Backend (*opsi_config*)
Unter */var/lib/opsi* liegen die Konfigurationsdateien des file 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.

5.1.6 opsi PAM Authentifizierung

opsi verwendet zur Authentifizierung der User diverse PAM-Module. Bisher wurden für verschiedene Distributionen verschiedene PAM-Module verwendet. In der folgenden Auflistung werden die eingesetzten PAM Module aufgelistet:

Standard: `common-auth`
 openSUSE / SLES: `sshd`
 CentOS und RedHat: `system-auth`
 RedHat 6: `password-auth`

Wie man aus der Liste erkennen kann, wurden diverse PAM-Konfigurationen verwendet, diese können sich aber je nach lokaler PAM Konfiguration wieder ändern. Da für diese Anpassungen immer ein Eingriff in den Code nötig war gibt es nun die Möglichkeit unter: `/etc/pam.d/` die Datei `opsi-auth` an zu legen und für opsi eine eigene PAM-Konfiguration zu hinterlegen. Wenn es diese Datei gibt, benutzt opsi automatisch diese Konfiguration.

Folgendes einfaches Beispiel soll das Verhalten verdeutlichen: Wenn Sie ein Debian/Ubuntu System betreiben und bei der Anmeldung am `opsi-configed` eine PAM-Fehlermeldung bekommen, obwohl mit den selben Benutzerdaten eine SSH Verbindung zum Server geöffnet werden kann, kann man die Datei `/etc/pam.d/opsi-auth` mit folgendem Inhalt erstellen:

```
@include sshd
```

Nach einem Neustart von `opsiconfd` benutzt opsi automatisch das `sshd-PAM`-Modul zur authentifizierung.

Anmerkung

Bitte beachten Sie, dass die Anwendung der ACL auf case-sensitive arbeitende Schnittstellen zurückgreift, wohingegen die Authentifizierung über PAM case-insensitiv geschehen kann. Dadurch kann der Fall eintreten, dass trotz erfolgreicher Authentifizierung keine Arbeit mit dem Service möglich ist, da die ACL dies verhindern.

5.1.7 Problem-Management

Sollten Probleme im Betrieb mit dem opsi-Server auftreten so sollten folgende Dinge überprüft und ausgeführt werden. Die Erfahrung zeigt, dass die Kombination der folgender Befehle, die meisten Probleme behebt.

- Erreichbarkeit und Auslastung vom `opsi-webservice` prüfen:
<https://<server-ip>:4447/info> per Browser aufrufen. Wenn dies schon nicht geht, weiter mit nächstem Schritt. Wenn diese Seite angezeigt wird: Prüfen ob die Auslastung vom `opsi-webservice` zu hoch ist. Zur Anzeige der Auslastungsgrafiken wird `rrdtool` mitsamt Python-Bindings benötigt. Bitte installieren Sie diese gegebenenfalls nach.
 Am besten auch den freien Speicherplatz auf dem Server überprüfen (Weiter unten auf der info-Seite).
- Prüfen ob die benötigten Daemons laufen, eventuell neustarten:

```
ps -ef | grep opsiconfd
ps -ef | grep opsipxeconfd
service opsiconfd restart
service opsipxeconfd restart
```

- Konfiguration neu einlesen:

```
opsi-setup --init-current-config
```

- Rechte auf die opsi-relevanten Dateien und Ordner neu setzen:

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

- Backend aufräumen:

```
opsi-setup --cleanup-backend
```

- Prüfen ob Samba läuft, eventuell neustarten:

```
ps -ef | grep mbd
```

Es sollte mindestens ein `nmbd` und mindestens ein `smbd` prozess laufen. Eventuell samba neustarten:

```
service samba restart
```

oder

```
service nmbd restart
service smbd restart
```

- Neusetzen des pcpatch-Passworts:

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

5.2 Hinweise zum Wechsel zu Samba 4

Mit dem Erreichen des stable-Status von Samba4 wurde die Entwicklungs- und Maintenancearbeiten für den Samba3-Zweig eingestellt. Als Folge daraus werden fast alle gängigen Linux-Distributionen (Client- und Server-Varianten) mit Samba4 statt Samba3 ausgestattet. Daraus ergeben sich einige Veränderungen, die in diesem Kapitel dokumentiert werden sollen.

Samba-Freigaben sind zentraler Bestandteil für die Funktion von opsi. Durch das "generelle" Update auf Samba4 gibt es einige Dinge zu beachten, die in folgenden Kapiteln kurz erläutert werden sollen.

Zunächst muss unterschieden werden, in welchem Betriebsmodus Samba ausgeführt wird. Eine besondere Eigenschaft von Samba4 ist die Möglichkeit einen vollwertigen ActiveDirectory-Kompatiblen Domain Controller zu betreiben. In diesem Betriebsmodus (der aus Vereinfachungsgründen in den folgenden Kapiteln als PDC-Modus bezeichnet wird) gibt es Restriktionen, die aus Kompatibilitätsgründen vom ActiveDirectory übernommen werden mussten. In der Regel sind die neuen Distributionen mit Samba4 ausgestattet, allerdings nur mit dem normalen Freigaben-Betriebsmodus. Eine vollwertige ActiveDirectory Domain zu betreiben, ist mit den Standardpaketen von den Distributionen in der Regel nicht möglich. Eine Ausnahme stellt hier der Univention Corporate Server dar, bei dem auch in den Standardpaketen der PDC-Modus integriert ist.

5.2.1 Die `/etc/opsi/opsi.conf`: `pcpatch` und `opsifileadmins`

Tipp

Die Restriktion, die in diesem Kapitel beschrieben wird, betrifft nur den PDC-Modus von Samba4.

Die klassische Installationsvariante mit dem Benutzer: `pcpatch` mit der primären Gruppe: `pcpatch` kann für Installationen mit Samba4 nicht eingehalten werden. Da Samba4 den grundlegenden Restriktionen von Active-Directory unterliegt, sind Gruppen mit der gleichen Bezeichnung wie User (wie in Unix/Linux üblich) nicht mehr erlaubt. Aus diesem Grund wurde für Samba4 Installationen eine neue Konfigurationsdatei eingeführt: `/etc/opsi/opsi.conf`, über die gesteuert wird, wie die Gruppe für den Samba-Zugriff auf die Freigaben bestimmt wird. Im Fall von Samba4 Installationen wird nun über diese Datei der Gruppenname `pcpatch` umbenannt und heißt von nun an: `opsifileadmins`. Das bedeutet, dass die User, die Zugriffsrechte für die Freigaben von opsi erhalten müssen (opsi-Paketierer) unter Samba4 nicht Mitglied der Gruppe `pcpatch` werden können, sondern Mitglied der Gruppe `opsifileadmins` sein müssen.

Weiterhin muss in diesem Fall der User `pcpatch` nun als vollwertiger Domänenbenutzer angelegt werden und nicht mehr als Systemuser, da er ansonsten auf die Domänenfreigaben nicht zugreifen kann.

Diese Schritte werden bei einer Installation von opsi auf einem Univention Corporate Server automatisch ausgeführt, wenn bei der Installation erkannt wird, dass das Samba4 im PDC-Modus läuft.

Da es außer den UCS-Installationen noch keine Standard-ActiveDirectory Konfiguration existiert, müssen diese Schritte bei einem manuell aufgesetzten Samba4 ActiveDirectory Domaincontroller manuell konfiguriert werden. Wenn das opsi System bei einer späteren Aktualisierung merkt, dass die User schon existieren, werden Sie bei der Aktualisierung nicht mehr angelegt.

Für Rückfragen kontaktieren Sie bitte den Support von opsi. Falls Sie keinen Supportvertrag haben, wenden Sie sich bitte an [info\(at\)uib.de](mailto:info(at)uib.de).

5.2.2 Freigaben-Konfiguration

Tipp

Die Änderungen, die in diesem Kapitel beschrieben werden betreffen alle Betriebsmodis von Samba4.

In Samba3 war es allgemein erlaubt, jede Datei oder Verzeichnis auf den Clients auszuführen. Dieses Verhalten wurde in Samba4 komplett verändert. Nun müssen alle Dateien, die über den Share ausführbar sein sollen, auch auf der Unix-Seite das Executable-Bit gesetzt haben.

Dies stellt ein allgemeines Problem für den Betrieb von opsi dar. Es ist nicht möglich dieses Verhalten über die Rechteverwaltung von opsi zu umgehen, da dies eine komplette Überarbeitung des Rechtesystems von opsi erfordern würde. Dies ist in opsi 4 nicht möglich.

Um das Problem mit opsi 4.0 dennoch zu umgehen, gibt es zwei Möglichkeiten.

Variante 1: auf den betroffenen Freigaben kann über die Freigabenkonfiguration über die Option: `admin users = @pccpatch`

für jedes Mitglied der pccpatch-Gruppe (Freigaben-User) dieses Verhalten ausgehebelt werden. Diesen Fix setzt opsi schon seit längerem auch bei UCS ≥ 3 mit Samba4 ein. Bei diesem Fix wird der Samba-Prozess der User mit erhöhten Rechten ausgeführt.

opsi setzt automatisch bei Samba4 Distributionen über `opsi-setup --auto-configure-samba` diese Option für den `opsi_depot` Share. Da dieser nur `readonly` gemounted wird, ist das Sicherheitsrisiko relativ gering.



Achtung

Für alle anderen Freigaben, die auch Read-Write gemounted werden können, bleibt zu bedenken, dass durch diesen Fix der Samba-Prozess mit erhöhten Rechten ausgeführt wird. Dies kann zu einer potentiellen Gefahr werden. Zur Zeit sind allerdings keine Exploits bekannt, die diesen Umstand als Schwachstelle ausnutzen würden, dennoch ist das natürlich keine Garantie, dass ein solcher Exploit nicht doch existiert.



Achtung

Der Linux smb Daemon hat einen Bug. Dieser steht in Kombination der `opsi_depot` Share-Definition in der `smb.conf`. Die `oplock` Parameter müssen bei bestehenden Installationen entfernt werden. Neue opsi-Installationen und dementsprechend neue Shares werden ohne `oplocks` angelegt.

Variante 2: Man kann global folgende Option in der `smb.conf` setzen: `acl allow execute always = true`

Durch diese Option wird für alle Freigaben das Verhalten von Samba 3 wiederhergestellt.

Sollte es gewünscht sein, dass die anderen Shares von diesem Problem nicht mehr betroffen sind, kann man entweder die erste Variante für jeden Share manuell setzen oder die Option von Variante zwei global in die `smb.conf` setzen. Die zweite Option gilt dann aber für alle Freigaben, auch für die, die nicht von und für opsi bereitgestellt werden.

Diese Variante funktioniert bei Univention Corporate Server nicht, da hier eine sehr stark angepasste Samba4 Variante eingesetzt wird.

5.2.3 Zugriff auf die Freigaben: `clientconfig.depot.user`

Diese Restriktion betrifft alle Betriebsmodis von Samba4.

Im Rahmen der Verwendung von Samba4 kann es notwendig sein zum mounten des depotshares explizit anzugeben mit welcher Domain / User Kombination dies erfolgen soll. Dazu gibt es den neuen config: `clientconfig.depot.user`. Gibt es diesen config nicht, so wird der user `pcpatch` genommen. Der Wert des config hat den Syntax: `<domain name>\<user name>` Ein config:

```
clientconfig.depot.user = opsiserver\pcpatch
```

gibt an, dass bei dem Mount des depotshares zur Authentifizierung als domain `opsiserver` und als user `pcpatch` angegeben werden soll. Die Erstellung eines solchen config kann über den opsi-configed erfolgen: Serverkonfiguration / `clientconfig` / Rechte Maustaste: Standard Konfigurationseintrag hinzufügen.

Die Erstellung eines solchen config kann auch auf der Kommandozeile erfolgen (wobei `pcpatch` durch den gewünschten string z.B. `opsiserver\pcpatch` ersetzt werden muß:

```
opsi-admin -d method config_createUnicode clientconfig.depot.user "clientconfig.depot.user" pcpatch
```

Dieser Systemweite config kann (z.B. im configed im Reiter Hostparameter) clientspezifisch angepasst werden.

5.3 opsi-Kommandozeilen-Werkzeuge und Prozesse

5.3.1 Werkzeug: `opsi-setup`

Das Programm `opsi-setup` ist das "Schweizer Taschenmesser" zur Einrichtung von opsi.

So greifen zum Beispiel die Pakete zur Installation von opsi auf dem Server auf dieses Skript zurück, um opsi korrekt einzurichten. Da dieses Skript nun auch extern aufrufbar ist, lassen sich damit diverse Wartungsarbeiten und Korrekturen durchführen:

- Registrierung eines opsi-servers als Depotserver
- Verzeichnisrechte korrigieren
- die Backends initialisieren
- Backends upgraden (von 3.4 nach 4.0)
- `mysql-Backend` erstmalig konfigurieren
- Default-Konfigurationen anpassen
- Backends bereinigen
- Samba-Konfiguration anpassen
- DHCP-Konfiguration anpassen

Hier die Hilfe-Anzeige von `opsi-setup`

```
opsi-setup --help

Usage: opsi-setup [options]

Options:
  -h, --help  show this help
  -l          log-level 0..9

  --log-file <path>      path to log file
  --ip-address <ip>     force to this ip address (do not lookup by name)
  --register-depot       register depot at config server
  --set-rights [path]   set default rights on opsi files (in [path] only)
  --init-current-config  init current backend configuration
  --update-mysql        update mysql backend
```

<code>--update-ldap</code>	update ldap backend
<code>--update-file</code>	update file backend
<code>--configure-mysql</code>	configure mysql backend
<code>--edit-config-defaults</code>	edit global config defaults
<code>--cleanup-backend</code>	cleanup backend
<code>--auto-configure-samba</code>	patch smb.conf
<code>--auto-configure-dhcpd</code>	patch dhcpd.conf

Die Funktionen und Parameter im Einzelnen:

- `--ip-address <ip>`
Diese Option dient dazu, eine IP-Adresse für den *opsi-server* vorzugeben und damit die Namensauflösung zu umgehen.
- `--register-depot`
Diese Option dient dazu, einen *opsi-server* an einem anderen *opsi-server* (*opsi-configserver*) als Depotserver anzumelden. Details hierzu vgl. Abschnitt [9.13.2](#).
- `--set-rights [path]`
Setzt bzw. korrigiert die Dateizugriffsrechte in den wesentlichen opsi-Verzeichnissen:
 - /tftpboot/linux
 - /home/opsiproducts
 - /var/log/opsi
 - /var/lib/opsi
 - /var/lib/opsi/depot
 - /etc/opsi
 Als Parameter kann auch ein Verzeichnis übergeben werden. Dann werden unterhalb dieses Verzeichnisses die Rechte aller opsi-relevanten Verzeichnisse und Dateien gesetzt, z.B.:
`opsi-setup --set-rights /var/lib/opsi/depot/winxppro/drivers`
- `--init-current-config`
Initialisiert die eingestellte Backendkonfiguration. Sollte nach jeder Änderung an der `/etc/opsi/backendManager/dispatch.conf` aufgerufen werden.
- Die Befehle `--update-mysql` und `--update-file`
dienen zum Upgrade des jeweiligen Backends (z.B. von opsi 3.4 auf opsi 4).
Details siehe *releasenotes-upgrade-Handbuch*.
- `--configure-mysql`
Dient zur erstmaligen Initialisierung des *mysql-Backend*.
Details vgl. Abschnitt [5.6.2](#).
- `--edit-config-defaults`
Zum Editieren der Defaultwerte der Config wie im *opsi-configed* in der Serverkonfiguration angezeigt.

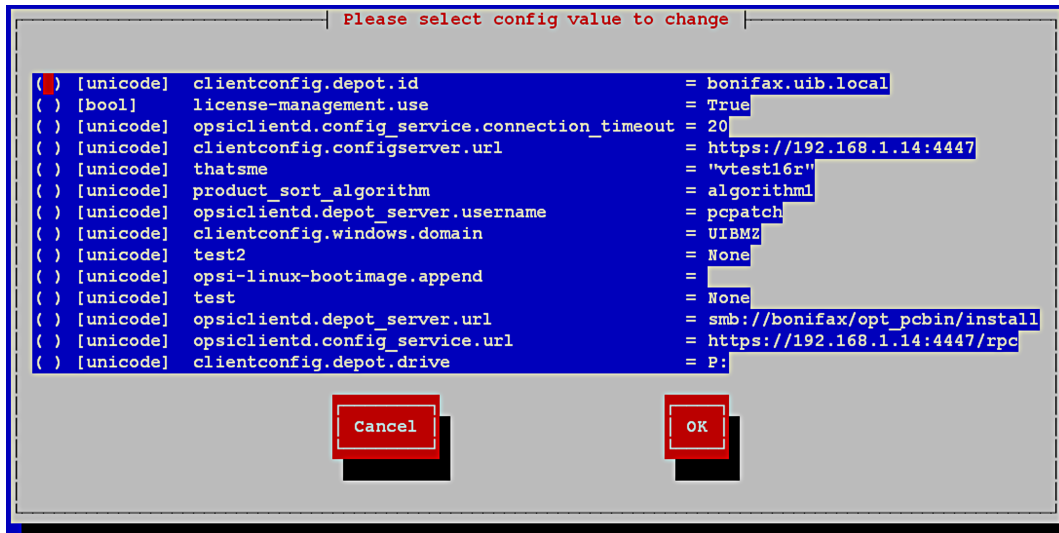


Abbildung 49: Eingabemaske: opsi-setup --edit-config-defaults

z.B.:

clientconfig.depot.id

Der Name des Default-Depotservers.

clientconfig.depot.drive

Der Laufwerksbuchstabe, auf welchem der Share mit den Installationsdaten verbunden wird. Hier kann ein Laufwerksbuchstabe fest gewählt werden oder über **dynamic** die automatische Auswahl ausgewählt werden. Bei dieser Variante wird versucht selbstständig einen freien Laufwerksbuchstaben zu finden.

license-management.use

Bestimmt, ob Netboot-Produkte Lizenzkeys aus dem Lizenzmanagement oder aus den Properties holen.

product_sort_algorithm

Bestimmt, nach welchem Algorithmus die Installationsreihenfolge ermittelt wird.

- **--cleanup-backend**

Überprüft die aktuellen Backends auf Integrität und verwirft obsolete oder unreferenzierte Einträge.

Entfernt werden dabei bspw. Produkte ohne Referenz (nicht auf Client / Depot installiert), Gruppen ohne Parent sowie ConfigStates ohne zugehörige Config.

Anmerkung

Es ist gängige, gute Praxis vor dem Aufräumen des Backends ein Backup durch *opsi-backup* zu erstellen.

- **--auto-configure-samba**

Erzeugt in der Samba-Konfigurationsdatei */etc/samba/smb.conf* die von opsi benötigten shares.

- **--auto-configure-dhcpd**

Erzeugt in der DHCP-Konfigurationsdatei */etc/dhcp3/dhcpd.conf* die von opsi benötigten Eintragungen.

Nur verwenden, wenn der DHCP-Server auf dem opsi Server laufen soll.

Details hierzu im DHCP-Kapitel des *opsi-getting-started* Handbuchs.

5.3.2 Werkzeug *opsi-package-manager*: opsi-Pakete (de-) installieren

Der *opsi-package-manager* dient zur (De-) Installation von Produkt-Paketen auf einem opsi-server.

Beim Aufruf von *opsi-package-manager* zur Installation muss das zu installierende Paket für den Systemuser *opsiconfd* lesbar sein. Es wird daher dringend empfohlen, Produkt-Pakete aus */home/opsiproducts* bzw. einem Unterverzeichnis hiervon zu installieren.

Die Logdatei des *opsi-package-manager* ist */var/log/opsi/package.log*

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-product-id myprod
```

Es ist möglich ein Paket mit einer abweichenden *Product ID* zu installieren. Das ist besonders dann hilfreich, wenn vorher ein Windows-Netboot-Produkt aus einem bestehenden Paket abgeleitet wurde und dieses Paket in der Zwischenzeit aktualisiert wurde.

```
opsi-package-manager --install win7-x64_1.2.3.opsi --new-product-id win7-x64-custom
```

Anmerkung

Bitte beachten Sie, dass *opsi-product-updater* derartig installierte Produkte nicht automatisch aktualisiert.

Eine Übersicht über alle Optionen liefert die Option `--help`.

Zu beachten:

- Die Optionen `-d` bzw. `--depots` sind für den Betrieb mehrerer Depotserver gedacht.
- Bei der Verwendung von `-d` wird das zu installierende Paket zunächst nach */var/lib/opsi/repository* kopiert. Dort muss ausreichend Platz zur Verfügung stehen. Siehe hierzu auch: *opsi-server* mit mehreren Depots Abschnitt [9.13](#)
- Falls es beim Installieren neuer Pakete zu Platzproblemen im temporären Verzeichnis kommt, kann mit der Option `--temp-dir` ein abweichender Ort angegeben werden.

```
# opsi-package-manager --help
```

```
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:
-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
--log-file-level <log-file-level> log file level (default 4)
-d, --depots <depots> comma separated list of depot ids to process
                    all = all known depots
-p, --properties <mode> mode for default product property values
                    ask      = display dialog
                    package = use defaults from package
                    keep     = keep depot defaults (default)
--purge-client-properties remove product property states of the installed product(s)
-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"
-o, --overwrite       overwrite existing package on upload even if size matches
-n, --no-delta        full package transfers on uploads (do not use librsync)
-k, --keep-files      do not delete client data dir on uninstall
-t, --temp-dir <path> temporary directory for package install
--max-transfers <num> maximum number of simultaneous uploads
                    0 = unlimited (default)
--max-bandwidth <kbps> maximum transfer rate for each transfer (in kilobytes per second)
                    0 = unlimited (default)
--new-product-id <product-id> Set a new product id when extracting opsi package or
                    set a specific product ID during installation.

```

5.3.3 Werkzeug: *opsi-product-updater*

Das Kommandozeilen Werkzeug *opsi-product-updater* dient dazu, komfortabel opsi-Produkte aus einem Repository zu laden und auf dem Server zu installieren. Daneben kann es auch per cronjob zeitgesteuert aufgerufen werden und so zur automatischen Synchronisation von opsi-Servern bzw. für automatische Updates verwendet werden.

```

# opsi-product-updater --help

Usage: opsi-product-updater [options]
Options:
-h      Show this help text
-v      Increase verbosity (can be used multiple times)
-V      Show version information and exit
-c      Location of config file
-i      Install all downloadable packages from configured repositories (ignores excludes)
-p      List of productIds that will be processed: opsi-winst,opsi-client-agent

```

Die wesentlichen Features sind *konfigurierbare Repositories* und *konfigurierbare Aktionen* (die Konfigurationseinstellungen werden in der `/etc/opsi/opsi-product-updater.conf` vorgenommen).

Konfigurierbare Repositories

Repositories sind die Quellen, von denen sich der opsi-server die Pakete holt.

Grundsätzlich gibt es zwei Arten von Repositories, *Internet-Repositories* und *opsi-Server*:

Internet-Repositories

Das wichtigste Beispiel ist das uib-Repository mit der URL <http://download.uib.de>

Internet-Repositories sind gekennzeichnet durch die Parameter

- *baseURL* (z.B. <http://download.uib.de>)
- *dirs* (Eine Liste von Verzeichnissen z.B. opsi4.0/produkte/essential)

- sowie bei Bedarf *username* und *password* für Passwort-geschützte Repositories (z.B. für Abo-Kunden)

Bei Bedarf ist auch ein Proxy einzustellen.

opsi-server

Ein Repository hat den Typ *opsi-server*, wenn in der `/etc/opsi/opsi-product-updater.conf` in der Sektion des Repositories ein Eintrag zum Punkt

- *opsiDepotId*

vorgenommen wird.

In der Regel ist bei einem *opsi-depotserver* an diese Stelle der zentrale *configserver* einzutragen. Damit zieht der Depotserver seine Pakete per Aufruf des *opsi-product-updater* bzw. automatisiert per Cronjob vom zentralen Server.

Konfigurierbare Aktionen

Für jedes Repository kann eingestellt werden:

- *autoupdate*: Aktuellere Versionen installierter Pakete werden geholt und installiert.
- *autoinstall*: Auch bis jetzt nicht installierte Pakete werden geholt und installiert
- *autosetup*: Die geholten und installierten Pakete werden für alle Clients, auf denen dieses Produkt installiert ist, auf *setup* gesetzt.
- *onlyDownload*: Neue Pakete werden nur heruntergeladen, es finden aber keine weiteren Aktionen damit statt. Ein beliebiger Anwendungsfall ist diese Option in Verbindung mit aktivierten Benachrichtigungen zu verwenden, so dass nach dem Download eine Mail versendet wird und die Installation zu einem späteren Zeitpunkt manuell durch einen Administrator erfolgt.

Zusätzlich ist es möglich, die Aktualisierung der Pakete auf den Clients über einen konfigurierbaren Wake-On-Lan-Mechanismus anzustoßen. In Verbindung mit dem Produkt *shutdownwanted* kann dafür gesorgt werden, dass die Clients nacheinander geweckt, die Software verteilt und die Clients danach wieder heruntergefahren werden. Hierdurch kann man seine Clients zum Beispiel außerhalb der Geschäftszeiten mit Updates und Software versorgen und die Anwender können am nächsten Morgen direkt mit der Arbeit beginnen.

5.3.4 Werkzeuge: opsi-admin / opsi config interface

Übersicht

Seit opsi 3.0 enthält eine serverseitige Bibliothek die zentralen Zugriffsfunktionen auf die opsi-Datenhaltung. Nach außen bietet sie eine API an, mit der ihre Funktionen genutzt werden können. Der *opsiconfd* stellt die komplette API als Webservice zur Verfügung.

Über den Aufruf von <https://<opsi-server>:4447/interface> kann über ein grafisches Frontend in elementare Form auf diesen Webservice zugegriffen werden. Dazu müssen Sie sich als Mitglied der Gruppe *opsiadmin* authentifizieren.

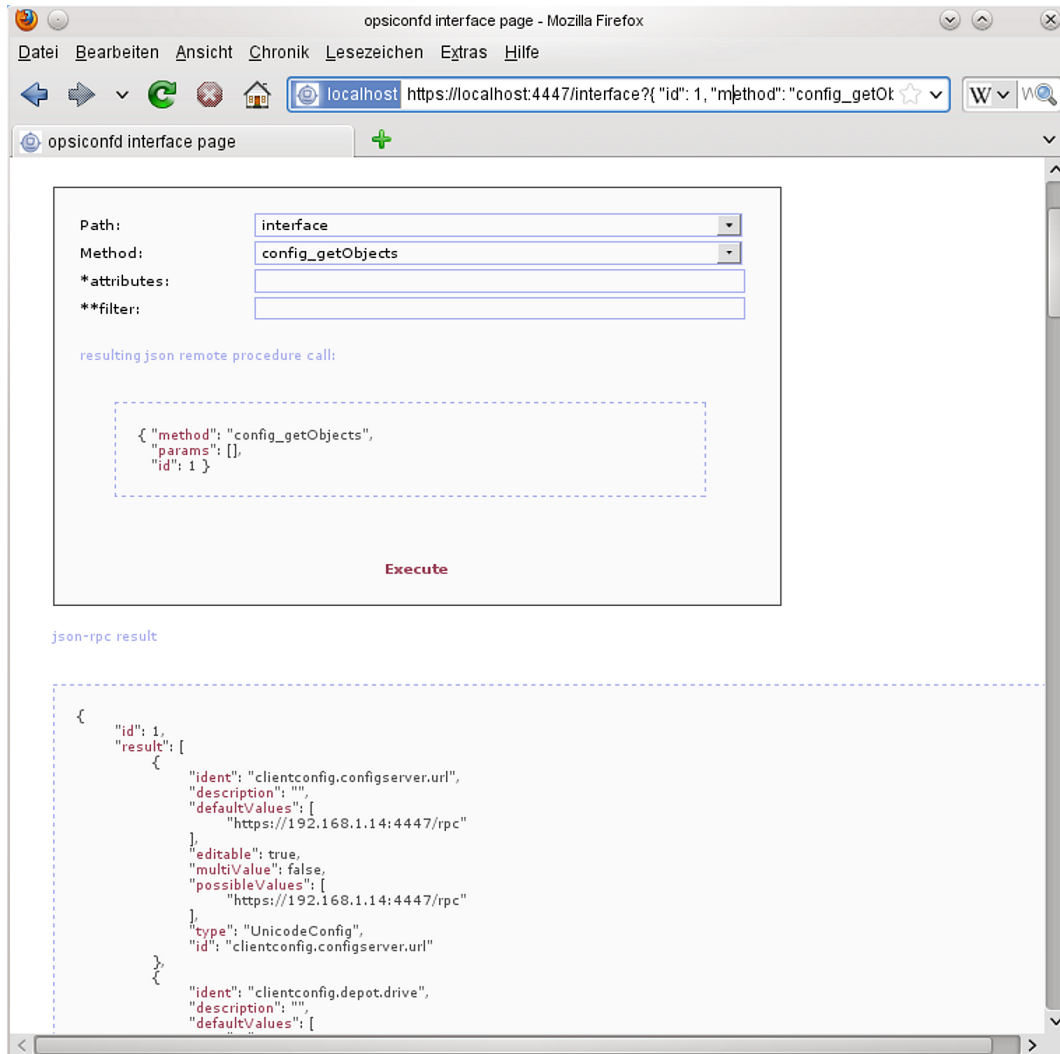


Abbildung 50: opsi-confd: Web-Interface

Auf der Kommandozeile kann mit dem Befehl `opsi-admin` auf die API zugegriffen werden. Dabei bietet *opsi-admin* einen interaktiven Modus und einen nicht-interaktiven z.B. zum Einsatz in Skripten.

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

```
# opsi-admin --help

Verwendung: opsi-admin [options] [command] [args...]
Optionen:
-h, --help           Diesen Hilfetext anzeigen
-V, --version        Versionsnummer ausgeben und beenden
-u, --username       Benutzername (standard: momentaner Benutzer)
-p, --password       Passwort (standard: Passwort interaktiv abfragen)
-a, --address        URL des opsi-confd (standard: https://localhost:4447/rpc)
-d, --direct         opsi-confd umgehen
--no-depot           Depotserver-Backend nicht verwenden
-l, --loglevel       Log-Level (standard: 3)
                    0=nichts, 1=essenziell, 2=kritisch, 3=Fehler, 4=Warnungen
                    5=Hinweise, 6=Informationen, 7=debug, 8=debug2, 9=vertraulich
-f, --log-file       Pfad zur Log-Datei
-i, --interactive    Im interaktiven Modus starten
-c, --colorize       Farbige Ausgabe
-S, --simple-output   Einfache Ausgabe (nur für Skalare und Listen)
```

<code>-s, --shell-output</code>	Shell-Ausgabe
<code>-r, --raw-output</code>	Rohdaten-Ausgabe

`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`, kurz `opsi-admin -dic`) erhalten Sie Eingabe-Unterstützung durch die Tabtaste. Betätigen der Tabtaste führt auf eine Auswahl der der möglichen Fortsetzungen der Eingabe bzw. die Angabe des Datentyps der nächsten erwarteten Eingabe. In der Liste der möglichen Eingaben können Sie mit Bild-auf und Bild-ab blättern.

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

Außer den Methodenaufrufen (eingeleitet mit `method`), welche direkt die API widerspiegeln, gibt es Aufrufe (eingeleitet mit `task`), die intern auf eine Kombination von Methodenaufrufen zur Erledigung einer bestimmten Aufgabe abgebildet werden.

Typische Verwendung

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

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

Liste aller Clients

```
opsi-admin -d method host_getIds
```

Client löschen

```
opsi-admin -d method host_delete <clientname>
```

z.B.:

```
opsi-admin -d method host_delete "pxevm.uib.local"
```

Client anlegen

```
opsi-admin -d method host_createOpsIClient <full qualified clientname>
```

z.B.:

```
opsi-admin -d method host_createOpsIClient "pxevm.uib.local"
```

Action Request setzen

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

z.B.:

```
opsi-admin -d method setProductActionRequest win7 pxevm.uib.local setup
```

Beschreibungen den Clients zuordnen

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

IDs aller Clients auflisten

Hierzu wird die Option `-S` verwendet, um zu erreichen, dass jeder Client in einer eigenen Zeile ausgegeben wird. Durch die Eingrenzung auf den Typ `OpsIClient` wird verhindert, dass opsi-Server mit ausgegeben werden.

Diese Ausgabe eignet sich zur Weiterverwendung in anderen Aufrufen.


```
opsi-admin -dS method host_getIdents '' '{"type": "OpsIClient"}'
```

Auflisten der auf Clients installierten Produkte

```
opsi-admin -d method productOnClient_getObjects '["productVersion", "packageVersion", "installationStatus"]' '{"\n  installationStatus": "installed"}'
```

Pcpatch-Passwort setzen

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

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

5.3.5 Serverprozesse: *opsiconfd* und *opsipxeconfd*

Der *opsipxeconfd* dient zur Bereitstellung von *named pipes* im *tftpboot*-Bereich, welche den Bootvorgang eines PCs über das PXE-Protokoll steuern.

Die zugehörige Konfigurationsdatei ist `/etc/opsi/opsipxeconfd.conf`, die Logdatei `/var/log/opsi/opsipxeconfd.log`.

Der *opsiconfd* dient zur Bereitstellung der opsi-server-API als JSON-Webservice und nimmt noch eine Reihe weiterer Aufgaben wahr.

Dieser Dienst ist damit der zentrale opsi-Dienst. Über ihn wird z.B. sämtliche Kommunikation zwischen den Clients und dem Server abgewickelt.

Von daher ist die Möglichkeit, diesen Prozess und seine Last zu überwachen, ein wichtiges Werkzeug.

opsiconfd*-Überwachung: *opsiconfd info

Unter der Webadresse <https://<opsi-server>:4447/info> erhalten Sie grafisch aufbereitete Informationen über den Lastverlauf des *opsiconfd* der letzten Stunde, des letzten Tages, des letzten Monats und des letzten Jahrs sowie weitere tabellarische Informationen.

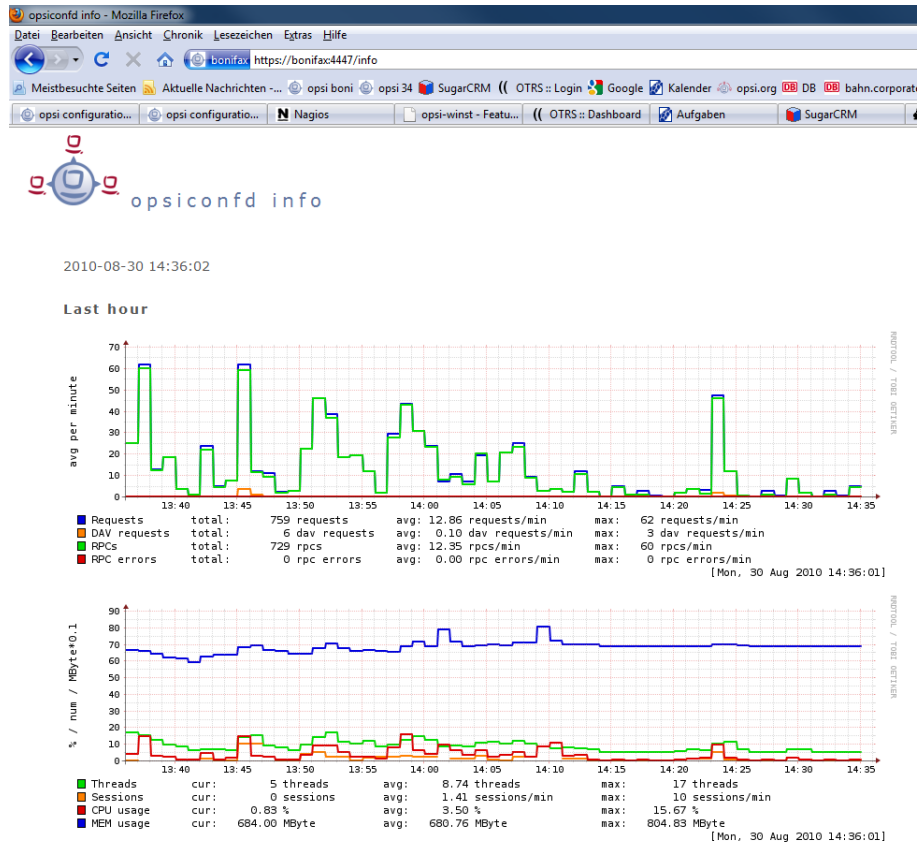


Abbildung 51: opsi-confd info: opsi-confd-Werte der letzten Stunde

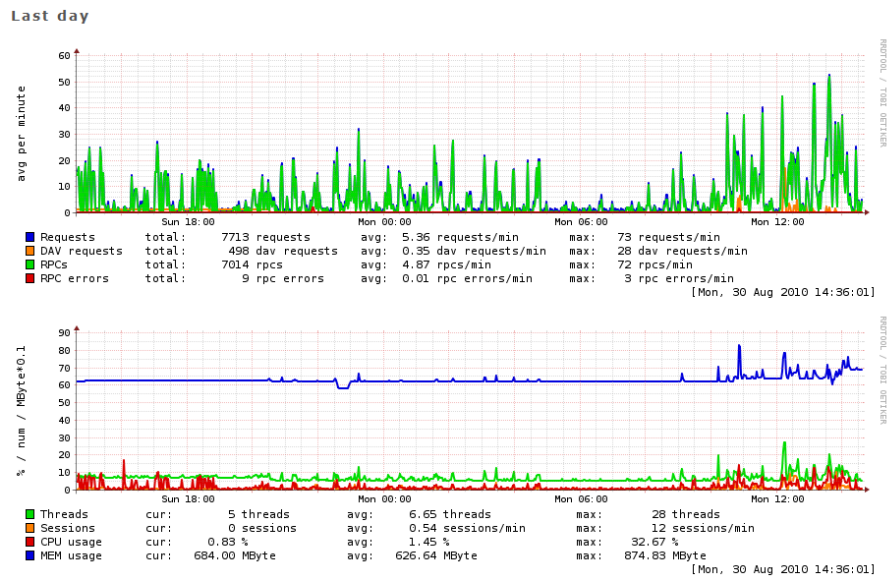


Abbildung 52: opsi-confd info: opsi-confd-Werte des letzten Tages

5.3.6 Serverprozess: opsi-atftpd

Der opsi-atftpd ist ein standard atftpd, welche um Fähigkeit erweitert wurde mit *named pipes* umzugehen.

Per default wird der opsi-atftpd so installiert, daß er nicht ständig läuft sondern nur bei Bedarf durch den `inetd` Prozess gestartet wird.

Um den opsi-atftpd als Daemon laufen zu lassen und ihn nicht über den `inetd` zu starten, müssen Sie folgende Änderungen vornehmen:

- In der `/etc/inetd.conf` die Zeile zum Protokoll `tftp` auskommentieren. Danach den `inetd` reloaden. Der Befehl hierzu ist abhängig vom verwendet Linux:
 - Ubuntu / Debian: `service openbsd-inetd reload`
 - openSUSE / SLES / RedHat / CentOS: `service xinetd reload`
- In der `/etc/default/atftpd` die Zeile `USE_INETD=true` ändern auf `USE_INETD=false`. Danach den opsi-atftpd als Daemon starten mit `service opsi-atftpd restart`

5.4 Web service / API Methoden

Seit opsi 4 gibt es zwei unterschiedliche Arten von API Methoden:

- *objekt orientierte* Methoden
- *aktions orientierte* Methoden

5.4.1 Web service / API Methoden seit opsi 4.0

Übersicht

Die opsi4-Backends basieren auf Objekten. Ein Objekt hat eine Reihe von Eigenschaften.

Als Beispiel diene hier das Objekt `product`. Das Objekt vom Typ `product`, welches das opsi-Produkt `javavm` beschreibt sieht z.B. so aus:

```
"ident": "javavm;1.6.0.20;2"
"id": "javavm"
"description": "Java&#x202f;1.6"
"changelog": ""
"advice": ""
"userLoginScript": ""
"name": "SunJavaRuntimeEnvironment"
"priority": 0
"packageVersion": "2"
"productVersion": "1.6.0.20"
"windowsSoftwareIds": None
"productClassIds": None
"type": "LocalbootProduct"
"licenseRequired": False
"setupScript": "javavm.ins"
"updateScript": ""
"uninstallScript": "deljvm.ins"
"alwaysScript": ""
"onceScript": ""
"customScript": ""
```

Zu jedem Objekt gibt es eine Reihe von Operationen. In der Regel sind dies:

- `getObjects` (liefert die Objekte)
- `getHashes` (Verwenden Sie besser `getObjects`)
- `create` (zum komfortablen Erzeugen eines Objektes)

- *createObjects* (zum Erzeugen vieler Objekte, Updatet vorhandene Objekte)
- *delete* (zum Löschen eines Objektes)
- *deleteObjects* (zum Löschen vieler Objekte. Es werden nur die das Object identifizierenden Merkmale zur Auswahl der zu löschenden Objekte verwendet.)
- *getIdents* (liefert nur die Objekt-Ids)
- *insertObject* (zum Erzeugen eines neuen Objektes, Updatet ein vorhandenes Objekt)
- *updateObject* (zum Aktualisieren eines Objektes, erzeugt **kein** Objekt, wenn nicht vorhanden. Daher besser *insertObject* verwenden.)
- *updateObjects* (zum Aktualisieren vieler Objekte, erzeugt **kein** Objekte, wenn nicht vorhanden. Daher besser *createObjects* verwenden.)

Die Namen der Methoden setzen sich zusammen aus:

`<object name>_<operation>`

Dadurch unterscheiden sie sich von den *Legacy* Methoden aus opsi 3.x welche in der Regel mit *get*, *set* oder *create* anfangen.

Die *getObjects*-Methoden haben zwei optionale Parameter:

- *attributes*
- *filter*

attributes dient dazu, nur bestimmte Attribute des Objektes abzufragen. Zurückgeliefert werden immer alle Attributnamen, aber nur die Werte der Attribute, welche das Objekt eindeutig kennzeichnen sowie die in *attributes* angegebenen Attributwerten. Die restlichen Attribute werden mit dem Wert *None* geliefert.

So liefert z.B die Methode *product_getObjects*, parametrisiert mit *attributes:["name"]* für das Produkt *javavm*:

```
"onceScript": None,
"ident": "javavm;1.6.0.20;2",
"windowsSoftwareIds": None,
"description": None,
"setupScript": None,
"changelog": None,
"customScript": None,
"advice": None,
"uninstallScript": None,
"userLoginScript": None,
"name": "Sun Java Runtime Environment",
"priority": None,
"packageVersion": "2",
"productVersion": "1.6.0.20",
"updateScript": None,
"productClassIds": None,
"alwaysScript": None,
"type": "LocalbootProduct",
"id": "javavm",
"licenseRequired": None
```

Wenn Sie keine *attributes*, aber einen *filter* angeben möchten, darf das Feld Sie für *attributes* nicht ganz leer bleiben, sondern muss den Wert `[]` erhalten.

Mit *filter* kann eingeschränkt werden, zu welchen Objekten Informationen geholt werden sollen, ähnlich einer sql-where-Bedingung. So schränkt für *product_getObjects* der Filter `{ "id": "javavm" }` die Rückgabe auf das Object *javavm* ein.

Bei den Methoden, denen ein oder mehrere Objekte übergeben werden, muss dies als JSON-Objekt bzw. als Liste von JSON-Objekten geschehen.

Die wichtigsten Objekte sind:

- *auditHardwareOnHost* (clientspezifische Hardwareinformationen)
- *auditHardware* (clientunabhängige Hardwareinformationen)
- *auditSoftwareOnClient* (clientspezifische Softwareinformationen)
- *auditSoftware* (clientunabhängige Softwareinformationen)
- *auditSoftwareToLicensePool* (Lizenzmanagement)
- *configState* (Verwaltung von Zusatzkonfigurationen)
- *config* (Verwaltung von neuen typisierten Zusatzkonfigurationen)
- *group* (Gruppenverwaltung)
- *host* (Server und Clients)
- *licenseContract* (Lizenzmanagement)
- *licenseOnClient* (Lizenzmanagement)
- *licensePool* (Lizenzmanagement)
- *objectToGroup* (Gruppenverwaltung)
- *productDependency* (Produktabhängigkeiten)
- *productOnClient* (Infos zu einem Produkt bezogen auf einen Client)
- *productOnDepot* (Infos zu einem Produkt bezogen auf ein Depot)
- *productPropertyState* (Depot und Client bezogene Product Property Werte)
- *productProperty* (Definition der Product Properties)
- *product* (Produkt Metadaten)
- *softwareLicenseToLicensePool* (Lizenzmanagement)
- *softwareLicense* (Lizenzmanagement)

Daneben gibt es noch eine Reihe von weiteren Objekten mit speziellen Operationen. Das aufgeführte Design ermöglicht es:

- schnell Informationen zu einer großen Zahl von Objekten zu übertragen,
- dabei mit einheitlicher Syntax Daten zu filtern,
- die Informationen auf syntaktische Korrektheit der erzeugten Objekte zu prüfen.

Hierdurch wird eine verbesserte Stabilität und höhere Performanz erreicht.

host (server und clients)

Beispiel für einen OpsiClient:

```
method host_getObjects [] {"id":"xpclient.vmnat.local"}
[
  {
    "ident" : "xpclient.vmnat.local",
    "description" : "",
    "created" : "2012-03-22 12:13:52",
    "inventoryNumber" : "",
    "ipAddress" : "172.16.166.101",
    "notes" : "Created by opsi-deploy-client-agent at Wed, 24 Aug 2011 10:24:36",
    "oneTimePassword" : "",
    "lastSeen" : "2012-03-30 16:20:04",
    "hardwareAddress" : "00:0c:29:35:70:a7",
    "opsiHostKey" : "1234567890abcef1234567890abcdef",
    "type" : "OpsiClient",
    "id" : "xpclient.vmnat.local"
  }
]
```

Die meisten dieser Daten finden sich im *clients* tab des opsi-configed.

Mögliche Werte für type:

- *OpsiClient*
- *OpsiConfigserver* (was bedeutet, dies ist auch ein *OpsiDepotserver*)
- *OpsiDepotserver*

Der Server type hat andere und mehr Daten als ein Client..

Beispiel für einen server:

```
method host_getObjects [] {"id":"sepiolina.vmnat.local"}
[
  {
    "masterDepotId" : null,
    "ident" : "sepiolina.vmnat.local",
    "networkAddress" : "172.16.166.0/255.255.255.128",
    "description" : "",
    "inventoryNumber" : "",
    "ipAddress" : "172.16.166.1",
    "repositoryRemoteUrl" : "webdavs://sepiolina.vmnat.local:4447/repository",
    "depotLocalUrl" : "file:///var/lib/opsi/depot",
    "isMasterDepot" : true,
    "notes" : "",
    "hardwareAddress" : null,
    "maxBandwidth" : 0,
    "repositoryLocalUrl" : "file:///var/lib/opsi/repository",
    "opsiHostKey" : "1234567890abcef1234567890abcdef",
    "type" : "OpsiConfigserver",
    "id" : "sepiolina.vmnat.local",
    "depotWebdavUrl" : "webdavs://sepiolina:4447/depot",
    "depotRemoteUrl" : "smb://sepiolina/opsi_depot"
  }
]
```

Die meisten dieser Daten finden sich in der *depot configuration* des opsi-configed.

group (Gruppen Verwaltung)

Beschreibt Gruppen und Ihre hierarchische Struktur

Beispiel für ein group Objekt:

```
method group_getObjects
[
  {
    "ident" : "sub2",
    "description" : "sub2",
    "notes" : "",
    "parentGroupId" : null,
    "type" : "HostGroup",
    "id" : "sub2"
  },
  {
    "ident" : "subsub",
    "description" : "subsub",
    "notes" : "",
    "parentGroupId" : "sub2",
    "type" : "HostGroup",
    "id" : "subsub"
  }
]
```

objectToGroup (Gruppen Verwaltung)

Beschreibt die Mitgliedschaft von Objekten in Gruppen.

Es gibt *Hostgroups* und *Productgroups*

Beispiel für ein objectToGroup Objekt:

```
method objectToGroup_getObjects
[
  {
    "groupType" : "HostGroup",
    "ident" : "HostGroup;sub2;win7.vmnat.local",
    "type" : "ObjectToGroup",
    "groupId" : "sub2",
    "objectId" : "win7.vmnat.local"
  },
  {
    "groupType" : "HostGroup",
    "ident" : "HostGroup;subsub;win7x64.vmnat.local",
    "type" : "ObjectToGroup",
    "groupId" : "subsub",
    "objectId" : "win7x64.vmnat.local"
  },
  {
    "groupType" : "ProductGroup",
    "ident" : "ProductGroup;opsiessentials;opsi-client-agent",
    "type" : "ObjectToGroup",
    "groupId" : "opsiessentials",
    "objectId" : "opsi-client-agent"
  },
  {
    "groupType" : "ProductGroup",
    "ident" : "ProductGroup;opsiessentials;opsi-winst",
    "type" : "ObjectToGroup",
    "groupId" : "opsiessentials",
    "objectId" : "opsi-winst"
  }
]
```

product (product meta data)

Beschreibt die Meta-Daten eines Produktes wie sie bei der Erstellung des Produktes definiert wurden.

Beispiel für ein product Objekt:

```
method product_getObjects [] {"id":"jedit","productVersion":"4.5"}
[
  {
    "onceScript" : "",
    "ident" : "jedit;4.5;3",
    "windowsSoftwareIds" :
      [
      ],
    "description" : "jEdit with opsi-winst Syntax-Highlighting",
    "setupScript" : "setup.ins",
    "changelog" : "",
    "customScript" : "",
    "advice" : "",
    "uninstallScript" : "uninstall.ins",
    "userLoginScript" : "",
    "name" : "jEdit programmer's text editor",
    "priority" : 0,
    "packageVersion" : "3",
    "productVersion" : "4.5",
    "updateScript" : "update.ins",
    "productClassIds" :
      [
      ],
    "alwaysScript" : "",
    "type" : "LocalbootProduct",
    "id" : "jedit",
    "licenseRequired" : false
  }
]
```

Anmerkung

Im Fall von mehreren Depotservern, können hier unterschiedliche Versionen eines product auftauchen.

Die Eintragungen für *productClassIds* und *windowsSoftwareIds* werden im Moment nicht verwendet.

productProperty (Definition der product properties)

Beschreibt die properties eines product wie sie bei der Erstellung des Produktes definiert wurden.

Beispiel für ein productProperty Objekt:

```
method productProperty_getObjects [] {"productId":"jedit","productVersion":"4.5"}
[
  {
    "ident" : "jedit;4.5;3;start_server",
    "description" : "Should the jedit derver started at every startup ?",
    "editable" : false,
    "defaultValues" :
      [
        false
      ],
    "multiValue" : false,
    "productVersion" : "4.5",
    "possibleValues" :
      [
        false,
        true
      ],
    "packageVersion" : "3",
    "type" : "BoolProductProperty",
  }
]
```



```

    "propertyId" : "start_server",
    "productId" : "jedit"
  }
]

```

Anmerkung

Die für einen Client verwendeten default Werte finden sich nicht hier, sondern werden Depotspezifisch in productPropertyState Objekten gespeichert.

productPropertyState (Depot oder Client spezifische product property settings)

Beschreibt:

- die default Werte eines product property auf einem Depot
- die Client spezifischen settings eines product properties.

Beispiel für ein productPropertyState Objekt:

```

method productPropertyState_getObjects [] {"productId":"jedit"}
[
  {
    "ident" : "jedit;start_server;sepiolina.vmnat.local",
    "objectId" : "sepiolina.vmnat.local",
    "values" :
      [
        false
      ],
    "type" : "ProductPropertyState",
    "propertyId" : "start_server",
    "productId" : "jedit"
  },
  {
    "ident" : "jedit;start_server;xpclient.vmnat.local",
    "objectId" : "xpclient.vmnat.local",
    "values" :
      [
        true
      ],
    "type" : "ProductPropertyState",
    "propertyId" : "start_server",
    "productId" : "jedit"
  }
]

```

productDependency (product Abhängigkeiten)

Beschreibt die Abhängigkeit eines Produktes zu einem anderen Produkt wie sie bei der Erstellung des Produktes definiert wurden.

Beispiel für ein productDependency Objekt:

```

method productDependency_getObjects [] {"productId":"jedit","productVersion":"4.5"}
[
  {
    "ident" : "jedit;4.5;3;setup;javavm",
    "productAction" : "setup",
    "requiredPackageVersion" : null,
    "requirementType" : "before",

```

```

    "requiredInstallationStatus" : "installed",
    "productVersion" : "4.5",
    "requiredProductId" : "javavm",
    "requiredAction" : null,
    "requiredProductVersion" : null,
    "type" : "ProductDependency",
    "packageVersion" : "3",
    "productId" : "jedit"
  }
]

```

productOnClient (client spezifische Informationen zu einem Produkt z.B. Installationsstatus)

Beschreibt welche Produkte in welchen Versionen auf welchem Client installiert sind.

Beispiel für ein productOnClient Objekt:

```

method productOnClient_getObjects [] {"productId":"jedit","clientId":"xpclient.vmnat.local"}
[
  {
    "ident" : "jedit;LocalbootProduct;xpclient.vmnat.local",
    "actionProgress" : "",
    "actionResult" : "successful",
    "clientId" : "xpclient.vmnat.local",
    "modificationTime" : "2012-03-30 15:49:04",
    "actionRequest" : "none",
    "targetConfiguration" : "installed",
    "productVersion" : "4.5",
    "productType" : "LocalbootProduct",
    "lastAction" : "setup",
    "packageVersion" : "3",
    "actionSequence" : -1,
    "type" : "ProductOnClient",
    "installationStatus" : "installed",
    "productId" : "jedit"
  }
]

```

productOnDepot (depot spezifische Informationen zu einem Produkt)

Beschreibt welches Produkt in welcher Version auf welchem Depot installiert ist.

Beispiel für ein productOnDepot Objekt:

```

method productOnDepot_getObjects [] {"productId":"jedit"}
[
  {
    "ident" : "jedit;LocalbootProduct;4.4.1;2;depotserver.vmnat.local",
    "locked" : false,
    "productVersion" : "4.4.1",
    "productType" : "LocalbootProduct",
    "depotId" : "depotserver.vmnat.local",
    "type" : "ProductOnDepot",
    "packageVersion" : "2",
    "productId" : "jedit"
  },
  {
    "ident" : "jedit;LocalbootProduct;4.5;3;sepiolina.vmnat.local",
    "locked" : false,
    "productVersion" : "4.5",
    "productType" : "LocalbootProduct",
    "depotId" : "sepiolina.vmnat.local",
    "type" : "ProductOnDepot",
    "packageVersion" : "3",
    "productId" : "jedit"
  }
]

```

```

    }
  ]

```

Anmerkung

Im Fall von mehreren Depotservern, können hier unterschiedliche Versionen eines Produktes auftauchen.

config (Verwaltung der Defaultwerte der Hostparameter)

Beschreibt die *Hostparameter* der *Server Konfiguration* des opsi-configeds.

Beispiel für ein config Objekt:

```

method config_getObjects [] {"id":"opsiclientd.event_gui_startup.active"}
[
  {
    "ident" : "opsiclientd.event_gui_startup.active",
    "description" : "gui_startup active",
    "defaultValues" :
      [
        true
      ],
    "editable" : false,
    "multiValue" : false,
    "possibleValues" :
      [
        false,
        true
      ],
    "type" : "BoolConfig",
    "id" : "opsiclientd.event_gui_startup.active"
  }
]

```

configState (Verwaltung der clientspezifischen Hostparameter)

Beschreibt die *Hostparameter* der *Client Konfiguration* des opsi-configeds..

Beispiel für ein configState Objekt:

```

method configState_getObjects [] {"configId":"opsiclientd.event_gui_startup.active"}
[
  {
    "configId" : "opsiclientd.event_gui_startup.active",
    "ident" : "opsiclientd.event_gui_startup.active;wanclient.vmnat.local",
    "values" :
      [
        false
      ],
    "objectId" : "wanclient.vmnat.local",
    "type" : "ConfigState"
  }
]

```

Anmerkung

Ein *configState* Objekt kann nicht erzeugt werden ohne das das *config* Objekt existiert auf das es referenziert.

auditHardwareOnHost (Clientspezifische Hardware Informationen)

Beschreibt die ermittelten Hardwaretypen (inclusive der clientspezifischen Daten). Die Idee ist in diesem Objekt die clientspezifischen Daten zu halten und in `auditHardware` nur die allgemeinen, so dass es dort z.B. nur einen Eintrag für eine Netzwerkkarte gibt, die in vielen Clients benutzt wird.

Leider funktioniert diese Idee in der Praxis nicht wirklich.

Das Attribut `state` legt fest ob die Daten aktuell (Wert = 1) oder historisch (Wert = 0) sind.

Beispiel für ein `auditHardwareOnHost` Objekt:

```
method auditHardwareOnHost_getObjects [] {"hostId":"xpclient.vmnat.local","hardwareClass":"NETWORK_CONTROLLER","\
ipAddress":"172.16.166.101"}
[
  {
    "vendorId" : "1022",
    "macAddress" : "00:0C:29:35:70:A7",
    "hardwareClass" : "NETWORK_CONTROLLER",
    "state" : 1,
    "deviceType" : "PCI",
    "subsystemVendorId" : "2000",
    "ipEnabled" : "True",
    "type" : "AuditHardwareOnHost",
    "firstseen" : "2012-03-30 15:48:15",
    "revision" : "10",
    "hostId" : "xpclient.vmnat.local",
    "vendor" : "Advanced Micro Devices (AMD)",
    "description" : "Ethernetadapter der AMD-PCNET-Familie",
    "subsystemDeviceId" : "1022",
    "deviceId" : "2000",
    "autoSense" : null,
    "netConnectionStatus" : "Connected",
    "maxSpeed" : null,
    "name" : "Ethernetadapter der AMD-PCNET-Familie",
    "serialNumber" : null,
    "lastseen" : "2012-03-30 15:48:15",
    "model" : null,
    "ipAddress" : "172.16.166.101",
    "adapterType" : "Ethernet 802.3"
  },
  {
    "vendorId" : "1022",
    "macAddress" : "00:0C:29:35:70:A7",
    "hardwareClass" : "NETWORK_CONTROLLER",
    "state" : 0,
    "deviceType" : "PCI",
    "subsystemVendorId" : "2000",
    "ipEnabled" : "True",
    "type" : "AuditHardwareOnHost",
    "firstseen" : "2012-03-08 14:26:14",
    "revision" : "10",
    "hostId" : "xpclient.vmnat.local",
    "vendor" : "VMware, Inc.",
    "description" : "VMware Accelerated AMD PCNet Adapter",
    "subsystemDeviceId" : "1022",
    "deviceId" : "2000",
    "autoSense" : null,
    "netConnectionStatus" : "Connected",
    "maxSpeed" : null,
    "name" : "VMware Accelerated AMD PCNet Adapter",
    "serialNumber" : null,
    "lastseen" : "2012-03-10 14:47:15",
    "model" : null,
    "ipAddress" : "172.16.166.101",
    "adapterType" : "Ethernet 802.3"
  },
  {
    "vendorId" : "1022",
```

```

    "macAddress" : "00:0c:29:35:70:a7",
    "hardwareClass" : "NETWORK_CONTROLLER",
    "state" : 0,
    "deviceType" : null,
    "subsystemVendorId" : "1022",
    "ipEnabled" : null,
    "type" : "AuditHardwareOnHost",
    "firstseen" : "2012-02-29 15:43:21",
    "revision" : "10",
    "hostId" : "xpclient.vmnat.local",
    "vendor" : "Advanced Micro Devices [AMD]",
    "description" : "Ethernet interface",
    "subsystemDeviceId" : "2000",
    "deviceId" : "2000",
    "autoSense" : "",
    "netConnectionStatus" : "yes",
    "maxSpeed" : null,
    "name" : "79c970 [PCnet32 LANCE]",
    "serialNumber" : "00:0c:29:35:70:a7",
    "lastseen" : "2012-03-30 14:58:30",
    "model" : "79c970 [PCnet32 LANCE]",
    "ipAddress" : "172.16.166.101",
    "adapterType" : ""
  }
]

```

auditHardware (Client unabhängige Hardware Informationen)

Beschreibt die ermittelten Hardwaretypen (ohne die clientspezifischen Daten). Die Idee ist in diesem Objekt nur die allgemeinen Daten eines Hardwaretyps zu halten, so dass es hier z.B. nur einen Eintrag für eine Netzwerkkarte gibt, die in vielen Clients benutzt wird.

Leider funktioniert diese Idee in der Praxis nicht wirklich.

Beispiel für ein `auditHardware` Objekt:

```

method auditHardware_getObjects [] {"hardwareClass":"NETWORK_CONTROLLER","vendorId":"1022"}
[
  {
    "vendorId" : "1022",
    "deviceId" : "2000",
    "maxSpeed" : null,
    "vendor" : "Advanced Micro Devices [AMD]",
    "name" : "79c970 [PCnet32 LANCE]",
    "subsystemDeviceId" : "2000",
    "deviceType" : null,
    "subsystemVendorId" : "1022",
    "autoSense" : "",
    "model" : "79c970 [PCnet32 LANCE]",
    "revision" : "10",
    "type" : "AuditHardware",
    "hardwareClass" : "NETWORK_CONTROLLER",
    "adapterType" : "",
    "description" : "Ethernet interface"
  },
  {
    "vendorId" : "1022",
    "deviceId" : "2000",
    "maxSpeed" : null,
    "vendor" : "VMware, Inc.",
    "name" : "VMware Accelerated AMD PCNet Adapter",
    "subsystemDeviceId" : "1022",
    "deviceType" : "PCI",
    "subsystemVendorId" : "2000",
    "autoSense" : null,
    "model" : null,
    "revision" : "10",
  }
]

```

```

    "type" : "AuditHardware",
    "hardwareClass" : "NETWORK_CONTROLLER",
    "adapterType" : "Ethernet 802.3",
    "description" : "VMware Accelerated AMD PCNet Adapter"
  },
  {
    "vendorId" : "1022",
    "deviceId" : "2000",
    "maxSpeed" : null,
    "vendor" : "Advanced Micro Devices (AMD)",
    "name" : "Ethernetadapter der AMD-PCNET-Familie",
    "subsystemDeviceId" : "1022",
    "deviceType" : "PCI",
    "subsystemVendorId" : "2000",
    "autoSense" : null,
    "model" : null,
    "revision" : "10",
    "type" : "AuditHardware",
    "hardwareClass" : "NETWORK_CONTROLLER",
    "adapterType" : "Ethernet 802.3",
    "description" : "Ethernetadapter der AMD-PCNET-Familie"
  },
  {
    "vendorId" : "1022",
    "deviceId" : "2000",
    "maxSpeed" : null,
    "vendor" : "Advanced Micro Devices (AMD)",
    "name" : "Ethernetadapter der AMD-PCNET-Familie",
    "subsystemDeviceId" : "1022",
    "deviceType" : "PCI",
    "subsystemVendorId" : "2000",
    "autoSense" : null,
    "model" : null,
    "revision" : "10",
    "type" : "AuditHardware",
    "hardwareClass" : "NETWORK_CONTROLLER",
    "adapterType" : "Ethernet 802.3",
    "description" : "Ethernetadapter der AMD-PCNET-Familie"
  },
  {
    "vendorId" : "1022",
    "deviceId" : "2000",
    "maxSpeed" : null,
    "vendor" : "Advanced Micro Devices (AMD)",
    "name" : null,
    "subsystemDeviceId" : "2000",
    "deviceType" : "PCI",
    "subsystemVendorId" : "1022",
    "autoSense" : null,
    "model" : "",
    "revision" : null,
    "type" : "AuditHardware",
    "hardwareClass" : "NETWORK_CONTROLLER",
    "adapterType" : null,
    "description" : "Ethernetadapter der AMD-PCNET-Familie"
  },
  (...),
  [

```

auditSoftwareOnClient (Clientspezifische Software Informationen)

Beschreibt die ermittelten Softwaretypen (inclusive der clientspezifischen Daten). Die Idee ist in diesem Objekt die clientspezifischen Daten zu halten und in `auditSoftware` nur die allgemeinen, so dass es dort z.B. nur einen Eintrag für eine Office-Software gibt, die in vielen Clients benutzt wird.

Beispiel für ein `auditSoftwareOnClient` Objekt:

```
method auditSoftwareOnClient_getObjects [] {"name":"jEdit 4.5.0","clientId":"xpclient.vmnat.local"}
[
  {
    "ident" : "jEdit 4.5.0;4.5.0;;;x86;xpclient.vmnat.local",
    "licenseKey" : "",
    "name" : "jEdit 4.5.0",
    "uninstallString" : "\\\"C:\\\\Programme\\\\jEdit\\\\unins000.exe\\\"",
    "usageFrequency" : -1,
    "clientId" : "xpclient.vmnat.local",
    "lastUsed" : "0000-00-00 00:00:00",
    "subVersion" : "",
    "language" : "",
    "state" : 1,
    "version" : "4.5.0",
    "lastseen" : "2012-03-30 16:19:55",
    "binaryName" : "",
    "type" : "AuditSoftwareOnClient",
    "firstseen" : "2012-03-30 16:19:55",
    "architecture" : "x86"
  }
]
```

auditSoftware (Client unabhängige Software Informationen)

Beschreibt die ermittelten Softwaretypen (ohne die clientspezifischen Daten). Die Idee ist in diesem Objekt nur die allgemeinen Daten eines Softwaretyps zu halten, so dass es hier z.B. nur einen Eintrag für eine Office-Software gibt, die in vielen Clients benutzt wird.

Beispiel für ein `auditSoftware` Objekt:

```
method auditSoftware_getObjects [] {"name":"jEdit 4.5.0"}
[
  {
    "windowsDisplayVersion" : "4.5.0",
    "ident" : "jEdit 4.5.0;4.5.0;;;x64",
    "name" : "jEdit 4.5.0",
    "windowsSoftwareId" : "jedit_is1",
    "windowsDisplayName" : "jEdit 4.5.0",
    "installSize" : -1,
    "subVersion" : "",
    "language" : "",
    "version" : "4.5.0",
    "architecture" : "x64",
    "type" : "AuditSoftware"
  },
  {
    "windowsDisplayVersion" : "4.5.0",
    "ident" : "jEdit 4.5.0;4.5.0;;;x86",
    "name" : "jEdit 4.5.0",
    "windowsSoftwareId" : "jedit_is1",
    "windowsDisplayName" : "jEdit 4.5.0",
    "installSize" : -1,
    "subVersion" : "",
    "language" : "",
    "version" : "4.5.0",
    "architecture" : "x86",
    "type" : "AuditSoftware"
  }
]
```

auditSoftwareToLicensePool (Lizenzmanagement)

Beschreibt die Zuordnung von Mustern aus der Softwareinventarisierung (`auditSoftware`) zu einzelnen Lizenzpools.

Beispiel für ein `auditSoftwareToLicensePool` Objekt:

```
method auditSoftwareToLicensePool_getObjects [] {"licensePoolId":"win7-msdn-prof"}
[
  {
    "ident" : "Windows 7 Professional N;6.1;00376-165;de-DE;x64;win7-msdn-prof",
    "name" : "Windows 7 Professional N",
    "language" : "de-DE",
    "subVersion" : "00376-165",
    "licensePoolId" : "win7-msdn-prof",
    "version" : "6.1",
    "architecture" : "x64",
    "type" : "AuditSoftwareToLicensePool"
  },
  {
    "ident" : "Windows 7 Professional N;6.1;00376-165;de-DE;x86;win7-msdn-prof",
    "name" : "Windows 7 Professional N",
    "language" : "de-DE",
    "subVersion" : "00376-165",
    "licensePoolId" : "win7-msdn-prof",
    "version" : "6.1",
    "architecture" : "x86",
    "type" : "AuditSoftwareToLicensePool"
  }
]
```

***softwareLicenseToLicensePool* (Lizenzmanagement)**

Beschreibt die Zuordnung von 'softwareLicenseId's zu 'licensePoolId's.

Beispiel für ein `softwareLicenseToLicensePool` Objekt:

```
method softwareLicenseToLicensePool_getObjects [] {"licensePoolId":"win7-msdn-prof"}
[
  {
    "licensePoolId" : "win7-msdn-prof",
    "softwareLicenseId" : "uib-msdn-win7-vol",
    "ident" : "uib-msdn-win7-vol;win7-msdn-prof",
    "licenseKey" : "12345-12345-12345-12345-3dbv6",
    "type" : "SoftwareLicenseToLicensePool"
  }
]
```

***softwareLicense* (Lizenzmanagement)**

Beschreibt die existierenden Softwarelizenzen und deren Metadaten.

Beispiel für ein `softwareLicense` Objekt:

```
method softwareLicense_getObjects [] {"id":"uib-msdn-win7-vol"}
[
  {
    "ident" : "uib-msdn-win7-vol;msdn-uib",
    "maxInstallations" : 0,
    "boundToHost" : null,
    "expirationDate" : "0000-00-00 00:00:00",
    "licenseContractId" : "msdn-uib",
    "type" : "VolumeSoftwareLicense",
    "id" : "uib-msdn-win7-vol"
  }
]
```


***licenseContract* (Lizenzmanagement)**

Beschreibt die existierenden Lizenzverträge und deren Metadaten.

Beispiel für ein `licenseContract` Objekt:

```
method licenseContract_getObjects [] {"id":"msdn-uib"}
[
  {
    "ident" : "msdn-uib",
    "description" : "",
    "conclusionDate" : "2011-04-22 00:00:00",
    "notificationDate" : "0000-00-00 00:00:00",
    "notes" : "",
    "expirationDate" : "0000-00-00 00:00:00",
    "partner" : "Microsoft",
    "type" : "LicenseContract",
    "id" : "msdn-uib"
  }
]
```

***licenseOnClient* (Lizenzmanagement)**

Beschreibt welcher Client welche Lizenz in Verwendung hat.

Beispiel für ein `licenseOnClient` Objekt:

```
method licenseOnClient_getObjects [] {"clientId":"win7client.vmnat.local"}
[
  {
    "softwareLicenseId" : "uib-msdn-win7-vol",
    "ident" : "uib-msdn-win7-vol;win7-msdn-prof;win7client.vmnat.local",
    "licenseKey" : "12345-12345-12345-12345-3dbv6",
    "notes" : "",
    "clientId" : "win7client.vmnat.local",
    "licensePoolId" : "win7-msdn-prof",
    "type" : "LicenseOnClient"
  }
]
```

***licensePool* (Lizenzmanagement)**

Beschreibt einen Lizenzpool und dessen Zuordnung zu Produkten.

Beispiel für ein `licensePool` Objekt:

```
method licensePool_getObjects [] {"id":"win7-msdn-prof"}
[
  {
    "ident" : "win7-msdn-prof",
    "type" : "LicensePool",
    "description" : "MSDN Keys",
    "productIds" :
      [
        "win7",
        "win7-x64"
      ],
    "id" : "win7-msdn-prof"
  }
]
```

Beispiel für die Änderung eines Keys in mehreren Objekten

Hier soll erläutert werden, wie Änderungen an einem Objekt durch geführt werden können. Als Beispiel wird das `host` Objekt verwendet, welches über die Auswahl auf den Typ `OpsiDepotserver` eingeschränkt wird:

```
method host_getObjects '[]' '{"type":"OpsiDepotserver"}'
[
  {
    "masterDepotId" : null,
    "ident" : "configserver.vmnat.local",
    "networkAddress" : "172.16.166.0/255.255.255.128",
    "description" : "",
    "inventoryNumber" : "",
    "ipAddress" : "172.16.166.1",
    "repositoryRemoteUrl" : "webdavs://configserver.vmnat.local:4447/reposit
tory",
    "depotLocalUrl" : "file:///var/lib/opsi/depot",
    "isMasterDepot" : true,
    "notes" : "",
    "hardwareAddress" : null,
    "maxBandwidth" : 0,
    "repositoryLocalUrl" : "file:///var/lib/opsi/repository",
    "opsiHostKey" : "17835c8d52170dcd06ba3c5089a74815",
    "type" : "OpsiConfigserver",
    "id" : "configserver.vmnat.local",
    "depotWebdavUrl" : "webdavs://configserver.vmnat.local:4447/depot",
    "depotRemoteUrl" : "smb://configserver/opsi_depot"
  },
  {
    "masterDepotId" : null,
    "ident" : "depotserver.vmnat.local",
    "networkAddress" : "172.16.166.128/25",
    "description" : "Depot Server",
    "inventoryNumber" : "",
    "ipAddress" : "172.16.166.150",
    "repositoryRemoteUrl" : "webdavs://depotserver.vmnat.local:4447/reposit
tory",
    "depotLocalUrl" : "file:///var/lib/opsi/depot",
    "isMasterDepot" : true,
    "notes" : "",
    "hardwareAddress" : "00:0c:29:7d:eb:55",
    "maxBandwidth" : 0,
    "repositoryLocalUrl" : "file:///var/lib/opsi/repository",
    "opsiHostKey" : "8284d506278667cb25cc2f9f992a024d",
    "type" : "OpsiDepotserver",
    "id" : "depotserver.vmnat.local",
    "depotWebdavUrl" : "webdavs://depotserver.vmnat.local:4447/depot",
    "depotRemoteUrl" : "smb://depotserver/opsi_depot"
  }
]
```

Zur Änderung der Werte für den Key "maxBandwidth" würde dieser Aufruf eine Datei erzeugen, in der die max. Bandbreite auf allen Depotservern von "0" auf "100" geändert wird. An der Datei können auch händisch Änderungen vorgenommen werden.

```
opsi-admin -d method host_getObjects '[]' '{"type":"OpsiDepotserver"}' | sed -e 's/"maxBandwidth":s:\s0/"maxBandwidth": 100/' > /tmp/maxBand.json
```

Hiermit wird die geänderte Konfiguration in das Opsi-Backend übernommen:

```
opsi-admin -d method host_createObjects < /tmp/maxBand.json
```

Spezielle Methoden

Es gibt eine Reihe von speziellen Methoden. Einige davon werden nachfolgend vorgestellt.

configState_getClientToDepotserver

Diese Methode liefert uns Informationen darüber welchem Depot ein Client zugordnet ist.

Die Syntax ist:

```
method configState_getClientToDepotserver *depotIds *clientIds
*masterOnly *productIds
```

Beispiel:

```
method configState_getClientToDepotserver [] "pcbon4.uib.local"
[
  {
    "depotId" : "bonifax.uib.local",
    "alternativeDepotIds" :
      [
      ],
    "clientId" : "pcbon4.uib.local"
  }
]
```

Kommunikation mit Hosts

Die *hostControl*-Methoden werden verwendet um mit den Clients zu kommunizieren und diese zu steuern. Seit opsi 4.0.3 empfohlen wird die Verwendung der *hostControlSafe*-Methoden. Beide Varianten haben einen Parameter *hostIds*. Bei *hostControl* ist dieser optional - ohne Angabe werden die Aktionen gegen alle Clients durchgeführt. Bei *hostControlSafe* ist der Parameter zwingend. Falls alle Clients angesprochen werden sollen, muss hier "*" angegeben werden. Bei Befehlen wie *hostControl_reboot* kann das Weglassen der Host-IDs zum versehentlichen Neustart aller Clients führen, weshalb in opsi 4.0.3 die Abwärtskompatibilität gebrochen wurde und bei den Befehlen *hostControl_reboot* und *hostControl_shutdown* das Weglassen des Parameters nun zu einem Fehler führt.

- **hostControlSafe_execute**
Führt einen Befehl auf den Clients aus.
Verbindet sich dazu mit dem opsiclientd auf den angegebenen Hosts und weist sie an **command** auszuführen.
Parameter: **command hostIds**
- **hostControlSafe_fireEvent**
Führt ein opsiclientd-Event auf den Clients aus. Verbindet sich dazu mit dem opsiclientd auf den angegebenen Hosts und weist ihn an das Event zu starten.
Parameter: **event hostIds**
- **hostControlSafe_getActiveSessions**
Liest die angemeldeten Benutzer aus.
Verbindet sich dazu mit dem opsiclientd auf den angegebenen Hosts und fragt die aktiven Sessions ab.
Parameter: **hostIds**
- **hostControlSafe_opsiclientdRpc**
Für die angegebene Methode auf dem Service des opsiclientd aus.
Verbindet sich dazu mit dem opsiclientd auf den angegebenen Hosts und führt die Methode **method** mit den gegebenen **params** als Parameter aus.
Das ist die generischste Methode, da sich hierüber beliebige weitere Methoden starten lassen.
Der einfachste Weg die verfügbaren Methoden zu erfahren ist sich auf das Control-Interface des Zielclients zu verbinden. Dieses ist zu erreichen unter <https://<clientId>:4441>.
Parameter: **method *params hostIds**
- **hostControlSafe_reachable**
Überprüft, ob der opsiclientd erreichbar ist.
Baut dazu eine Verbindung zum opsiclientd der gegebenen Clients auf, aber führt keinen Login durch.
Parameter: **hostIds**

- **hostControlSafe_reboot**
Startet die Clients neu.
Verbindet sich dazu mit dem `opsiclientd` auf den angegebenen Hosts und führt einen Neustart aus.
Parameter: `hostIds`
- **hostControlSafe_showPopup**
Zeigt eine Nachricht in einem Popup auf den Clients.
Verbindet sich dazu mit dem `opsiclientd` auf den angegebenen Hosts und zeigt die gegebene Nachricht in einem Popup-Fenster.
Parameter: `message hostIds`
- **hostControlSafe_shutdown**
Führt die Clients herunter.
Verbindet sich dazu mit dem `opsiclientd` auf den angegebenen Hosts und fährt den Client herunter.
Parameter: `hostIds`
- **hostControlSafe_start**
Sendet ein Wake-On-Lan-Signal an die Clients.
Das ist die einzige *hostControlSafe*-Methode, welche nicht den `opsiclientd` eines Clients verwendet.
Parameter: `hostIds`
- **hostControlSafe_uptime**
Fragt Clients nach ihrer Uptime. Verbindet sich dazu mit dem `opsiclientd` auf den angegebenen Hosts und liest die Uptime in Sekunden aus.
Parameter: `hostIds`

Arbeit mit Logs

Die folgenden Methoden drehen sich um die Arbeit mit Logs.

- **log_read**
Liest ein opsi-Log vom Server.
Parameter: `logType *objectId *maxSize`
Mögliche Log-Arten sind *instlog* (opsi-winst), *clientconnect* (opsiclientd), *userlogin*, *bootimage* sowie *opsiconfd*.
Als Parameter *objectId* wird normalerweise die *clientId* des Clients dessen Log man möchte angeben.
- **log_write**
Schreibt eine Logdatei zum Server.
Parameter: `logType data *objectId *append`
Log-Arten und *objectId* werden unter **log_read** beschrieben.
Mittels `append` wird gesteuert, ob der neue Inhalt an ein eventuell bestehendes Logfile angehängt wird. Der Wert kann `true` oder `false` sein. Letzteres ist der Standard.

Tutorial: Arbeit mit Gruppen

Sie arbeiten mit Gruppen-Objekten. Die Methoden beginnen also mit *group*. Dann gibt es zwei Arten von Gruppen: Host-Gruppen und Produkt-Gruppen. Für die Treeview benötigen wir die erste Variante, das heißt beim Erstellen oder Abfragen muss `type` auf `HostGroup` gesetzt werden.

Das Erstellen von Hostgruppen ist vereinfacht durch die Methode `group_createHostGroup`. Die Parameter sind `id`, `description` (Beschreibung), `notes` (Notizen) und `parentGroupId` (ID der übergeordneten Gruppe). Davon ist nur die `id` zwingend zu vergeben und diese muss einzigartig sein.



Wichtig

In opsi 4.0 werden Gruppen anhand ihrer ID identifiziert. Diese ID muss gruppenübergreifend einzigartig sein.

Das Anlegen einer Gruppe ist nun so möglich:

```
opsi-admin -d method group_createHostGroup rechner_wenselowski "Nikos Rechner"
```

Wollen Sie nun die angelegte Gruppe anschauen, so können Sie mittels `group_getObjects` die Gruppen abfragen. Hier die Abfrage nach der soeben angelegten Gruppe:

```
opsi-admin -d method group_getObjects '' '{"id": "rechner_*", "type": "HostGroup}"'
```

Wollen Sie Untergruppen anlegen, so müssen Sie als `parentGroupId` die ID der übergeordneten Gruppe angeben. Hier als Beispiel eine Untergruppe zur gerade angelegten Gruppe:

```
opsi-admin -d method group_createHostGroup "rechner_wenselowski2" "Untergruppe" "" "rechner_wenselowski"
```

Die Abfrage von vorher sollte nun auch die neue Gruppe ausgeben.

Wenn Sie nun mit dem Directory arbeiten wollen, so wird dieses intern als Gruppe mit der ID `clientdirectory` behandelt. Clients dürfen im Directory immer nur in einer Gruppe sein - per Default sind sie der Gruppe mit der ID `NICHT_ZUGEWIESEN` zugewiesen. Verwenden Sie opsi in einer anderen Sprache, kann der Gruppenname abweichen. Dafür, dass die Clients nur immer in einer Gruppe sind, wenn sie im Directory sein sollen, muss vom jeweiligen Programmierer gemanaged werden, da das Backend an dieser Stelle nicht eingreift.

Die Zuordnung von Clients zur Gruppe geschieht über `objectToGroup`-Objekte. Wir legen einen Client mit dem folgenden Befehl an:

```
opsi-admin -d method host_createOpsiClient "wenselowski-test.uib.local"
```

Diesen fügen wir nun der Untergruppe von vorhin hinzu:

```
opsi-admin -d method objectToGroup_create "HostGroup" "rechner_wenselowski2" "wenselowski-test.uib.local"
```

Um das nun zu überprüfen, können wir wie folgt die Zuordnung abfragen:

```
opsi-admin -d method objectToGroup_getObjects '' '{"groupType": "HostGroup", "groupId": "rechner_wenselowski2}"'
```

Um den Client aus der Gruppe zu entfernen, können Sie so vorgehen:

```
opsi-admin -d method objectToGroup_delete "HostGroup" "rechner_wenselowski2" "wenselowski-test.uib.local"
```

Um zu guter letzt eine Gruppe zu löschen, können Sie das wie folgt machen:

```
opsi-admin -d method group_delete "rechner_wenselowski"
```

5.4.2 Aktions-orientierte Methoden

Die mit opsi 3 eingeführten *aktions orientierten* Methoden stehen weiterhin zur Verfügung und werden weiter gepflegt. Diese Methoden werden aber ab opsi 4.0 intern auf die *objekt orientierten* Methoden *gemappt*.

Hier eine Liste der Methoden (dargestellt in der Form des Aufrufs mit *opsi-admin*) 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 authenticated
```

Überprüfen ob die Authentifizierung am Service erfolgreich war.

```
method createClient clientName domain
```

Erzeugt einen neuen Client.

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

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

```
method createLicenseKey productId licenseKey
```

Weist dem Produkt *productId* 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 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 possibleValues=[]
```

Erstellt eine Produkteigenschaft.

```
method deleteClient clientId
```

Löscht einen Client.

```
method deleteGeneralConfig objectId
```

Löscht Konfiguration eines Clients oder einer Domain.

```
method deleteGroup groupId
```

Löscht eine Clientgruppe.

```
method deleteHardwareInformation hostId
```

Löscht sämtliche Hardwareinfos zum Rechner *hostid*.

```
method deleteLicenseKey productId licenseKey
```

Löscht einen Lizenzkey.

```
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 getClientIds_list
```

Liefert die Liste der Clients, welche den angegebenen Kriterien entsprechen.

```
method getClients_listOfHashes
```

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

```
method getDomain hostId
```

Liefert 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 auditHardwareOnHost_getObjects '[]' '{"hostId":"<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 Host-ID.

```
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 Host-ID.

```
method getLicenseKey productId clientId
```

Liefert einen freien Lizenzkey zu dem angegebenen Produkt bzw. liefert den der *clientId* zugeordneten Lizenzkey,

```
method getLicenseKeys_listOfHashes productId
```

Liefert eine Liste der Lizenzkeys für das angegebene Produkt.

```
method getLocalBootProductIds_list
```

Liefert alle bekannten Localboot-Produkte.

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

Liefert für die angegebenen Clients Installationsstatus und Action-Requests 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 = []
```

Liefert für die angegebenen Clients Installationsstatus und {Action-request=} 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 Host-ID.

```
method getPcpatchPassword hostId
```


Liefert das mit dem *pkey* von *hostId* verschlüsselte Passwort des Users *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=softprod
```

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

```
method getPossibleProductInstallationStatus_list
```

Liefert die möglichen Installationsstatus (*installed, not_installed,...*).

```
method getPossibleRequirementTypes_list
```

Liefert die möglichen Typen von Produktabhängigkeiten (*before, after,...*).

```
method getProductActionRequests_listOfHashes clientId
```

Liefert die anstehenden ausführbaren Aktionen für den angegebenen Client.

```
method getProductDependencies_listOfHashe
```

Liefert die bekannten Produktabhängigkeiten (zum angegebenen Produkt).

```
method getProductIds_list
```

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-Properties) zum angegebenen Produkt und Client.

```
method getProductPropertyDefinitions_hash
```

Liefert alle bekannten Product-Properties mit Beschreibung, erlaubten Werten etc..

```
method getProductPropertyDefinitions_listOfHashes productId
```

Liefert die Product-Properties zum angegebenen Produkt mit Beschreibung, erlaubten Werten etc..

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

Liefert Installationsstatus und Action-Requests der einzelnen Produkte (zu den angegebenen 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-configserver zum angegebenen Client.

```
method getServerIds_list
```

Liefert die Liste der bekannten opsi-configserver.

```
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 setGeneralConfig config
```

Setzt für Client oder Domain die GeneralConfig.

```
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 setOpsiHostKey hostId opsiHostKey
```

Setzt für einen Rechner den *pkey*.

```
method setPcpatchPassword hostId password
```

Setzt das verschlüsselte (!) *password* für *hostId*.

```
method setProductActionRequest productId clientId actionRequest
```

Setzt für den angegebenen Client und das angegebene Produkt einen Action-Request.

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

Setzt für den angegebenen Client und das angegebene Produkt einen Installationsstatus.

```
method setProductProperties productId properties
```

Setzt Product-Properties für das angegebene Produkt (und den angegebenen Client).

```
method unsetProductActionRequest productId clientId
```

Setzt einen Action-Request auf *none*.

5.4.3 Backend-Erweiterungen

Durch die in opsi 4 implementierten Funktionalität des Backend-Extenders können auf der Basis der opsi4-API-Methoden oder des Funktionskerns jederzeit zusätzliche Methoden definiert und als API-Erweiterung aufrufbar gemacht werden.

Das Standard-API-Methoden-Set wird durch den *opsiconfd* mittels Überlagerung der in den Python-Dateien in */etc/opsi/backendManager/extend.d* definierten Methoden erstellt.

Zusätzliche Methoden-Sets mit Aufruf per speziellem Pfad können zudem in Dateien in Unterverzeichnissen dieses Verzeichnisses definiert werden. Standardmäßig ist hier ein Set von Methoden, die speziell auf Datenanforderungen durch den *opsi-configed* zugeschnitten sind, im Verzeichnis */etc/opsi/backendManager/extend.d/configed* implementiert. Im Web-Interface werden die entsprechenden Methoden (zusammen mit den Standardmethoden) sichtbar, wenn als *Path* in der Drop-down-Liste *interface/extend/configed* ausgewählt wird.

Backend-Erweiterungen können auch dazu dienen für spezifische Konfigurationsaufgaben angepasste Zusatzfunktionen zu implementieren.

Eigene Erweiterungen werden in der Programmiersprache Python geschrieben. Sie werden **an** den BackendManager geladen, so dass dieser mittels **self** referenziert werden kann.

5.4.4 Zugriff auf die API

Die API nutzt [JSON-RPC 1.0](#) über HTTP zur Kommunikation. Es wird *basic authentication* verwendet.

Um die Schnittstelle zu verwenden müssen Aufrufe per *POST* zum Pfad `rpc` am opsi-Server gesendet werden, bspw. `https://opsiserver.domain.local:4447/rpc`.

Achtung



Bei Kommunikation mit einem opsi 4.0 Webservice, liefert dieser einen HTTP-Header *content-type* welcher nicht zum tatsächlichen Inhalt passt. Es ist möglich ein zu [RFC 2616](#) kompatibles Verhalten zu aktivieren, indem die Datei `/etc/opsi/opsi.header.fix.enable` angelegt und der Service *opsiconfd* neu gestartet wird. Dadurch wird es einfacher für Software von Drittherstellern mit dem Webservice zu kommunizieren.

Da dieses Verhalten möglicherweise dazu führt, dass Clients, welches das alte Verhalten erwarteten, nicht mehr mit dem Service kommunizieren können, ist dies standardmäßig nicht aktiviert. Die im Rahmen von opsi 4.0.6 bereitgestellten Client-Komponenten wurden entsprechend aktualisiert, so dass Sie mit beiden Varianten umgehen können.

5.5 opsi-backup

5.5.1 Einführung

Wie jedes andere System auch, sollte das opsi-System auch einem Backup unterzogen werden. Da opsi ein zentrales Werkzeug für das Windows-Client- wie auch das Windows-Server-Management darstellt, sollte der opsi-server gesichert werden. Dieses Handbuch soll einen Einblick in die Backup-Strategie von opsi geben und auch auf Themen, wie das zurückschreiben und das "DisasterRecovery" von opsi.

5.5.2 Vorbedingungen für ein Backup

Um ein Backup des opsi-Systems anzulegen, gibt es nicht wirklich eine Vorbedingung. Wenn man die zentralen Dateien und Backends des opsi-Systems lokalisiert hat, kann man diese auf diversen Methoden sichern. Die folgende Anleitung soll nicht nur die Frage: "Was soll gesichert werden?" beantworten, sondern auch einen Weg dokumentieren, wie eine Backupstrategie für das opsi-System aussehen könnte.

Das Backupskript sollte als root ausgeführt werden, entweder manuell oder einen root-cronjob, damit man die Konfiguration von opsi lesen kann und auch die Systemkonfiguration feststellen kann. Weiterhin sollte für ein Backup des mysql-Backends das *mysqldump*-Programm installiert sein, dieses findet sich in der Regel in den client-Paketen von mysql.

5.5.3 Quick Start

Backup erzeugen:

```
opsi-backup create opsi_backup.tar.bz2
```

Erzeugt ein Backup der aktuell genutzten Backends sowie der Konfigurationsdateien im aktuellen Verzeichnis mit dem Namen `opsi_backup.tar.bz2`.

Backup zurück spielen:

```
opsi-backup restore --backends=all --configuration opsi_backup.tar.bz2
```

Stellt die Daten aus dem Backupfile `opsi_backup.tar.bz2` aus dem aktuellen Verzeichnis wieder her.

5.5.4 Elementare Teile von opsi

Opsi kann man grob in fünf Teile gliedern. Die folgenden fünf Teile sind opsi spezifisch und können von System zu System, je nach Konfiguration variieren.

Opsi Konfiguration

Der mit Abstand wichtigste Teil von opsi, ist die Konfiguration. Getreu nach LSB (Linux Standard Base) befindet sich die Konfiguration von opsi unter `/etc/opsi`.

Dieses Verzeichnis beinhaltet hauptsächlich die Backend-Konfiguration, die Webservice-Konfiguration und das SSL-Zertifikat für den Webservice. Weiterhin ist sind hier Backend-Erweiterungen untergebracht, die Konfigurationen des *opsipxeconfd*, des *opsi-product-updater* mit seinen Repositories und auch die modules-Datei, die Ihnen Ihre kofinanzierten Module freischaltet.

Um später eine volle Wiederherstellung nach einem Unglück zu verwirklichen, muss das Verzeichnis `/etc/opsi` gesichert werden.

Dieser Bereich wird mit `opsi-backup` gesichert.

Die Sicherung hat daneben noch einen weiteren Vorteil:

Wenn man viele Konfigurationen von opsi geändert hat und das System nicht mehr richtig arbeitet, ist ein Rücksprung auf eine vorherige funktionierende Version meist leichter und schneller, als die Fehlersuche.

Opsi Backends

Im folgenden Kapitel werden die Backends von opsi aufgezählt. Diese bilden das Herzstück der opsi-Datenhaltung. Alle Clients, Produkte, Konfigurationen, Statis, etc. . . sind in der jeweiligen Datenhaltung abgelegt.

Opsi bietet folgende Datenbackends:

Tabelle 1: opsi-Backends

Backend	Beschreibung
file-Backend	Backend auf Dateibasis, momentan der default bei opsi
mysql-Backend	Volle mysql-Backend Unterstützung seit opsi 4.0
dhcp	spezial Backend bei Verwendung von des dhcpd auf dem opsi-server

Wenn Sie nicht wissen, welches Backend sie einsetzen, setzen Sie wahrscheinlich das file-Backend ein. opsi ist aber auch dafür ausgelegt, mehrere Backends gleichzeitig an zu setzen. Welche Backends, für welche Funktionen von opsi eingesetzt werden, wird in der `/etc/opsi/backendManager/dispatch.conf` konfiguriert.

Dieser Bereich wird mit `opsi-backup` gesichert.

Opsi Depotfiles

Die Depotfiles sind deshalb interessant, da Sie die eigentlichen Dateien der zu verteilenden Software enthalten. Die Localboot-Produkte, wie auch die Netboot-Produkte haben Ihre Files jeweils unterhalb von `/var/lib/opsi/depot`. In früheren Versionen von opsi waren diese im Verzeichnis `/opt/pcbin/install` angesiedelt.

Je nachdem, wie viel Software auf dem opsi-server vorgehalten wird und wie viele Betriebssystem-Installationen inklusive Treibern vorgehalten werden, kann dieses Datenvolumen enorme Ausmaße annehmen.

Es gibt verschiedene Ansätze diese Dateien zu sichern. Die einfachste Alternative ist das *Rsnapshot*. Es gibt aber elegantere Lösungen, wie das Verlegen dieser Daten in redundant ausgelegte Filesysteme auf einem SAN, etc.

Dieser Bereich wird mit `opsi-backup` **nicht** gesichert.

Opsi Workbench

Der Bereich opsi Workbench, welcher auch als gleichnamige Samba-Freigabe (*opsi_workbench*) in opsi eingesetzt wird, beinhaltet die Stände der eigenen Software-Paketierung. Das Verzeichnis ist standardmäßig unter `/home/opsiproducts`, unter openSUSE und SLES ist dieses Verzeichnis unter `/var/lib/opsi/workbench`. Wenn dieser Share, wie vorgesehen dafür verwendet wird, um eigene Pakete in verschiedenen Revisionen dort vor zu halten, sollte dieses Verzeichnis auch gesichert werden.

Auch hier bietet sich das Tool *rsnapshot* an.

Dieser Bereich wird mit `opsi-backup` **nicht** gesichert.

Opsi Repository

Das Verzeichnis unter `/var/lib/opsi/repository` wird dazu verwendet, um opsi-Pakete zu puffern. Anders als die opsi Workbench, dient es aber nicht dem Paketieren von opsi Paketen, sondern die opsi Pakete welche dort abgelegt werden, sollen vorgehalten werden, um eventuell das Synchronisieren auf anderen Servern, oder das Synchronisieren mit dem `opsi-product-updater` zu vereinfachen.

Diese Dateien sind für ein vollständige Wiederherstellung nicht unbedingt nötig, können aber auch mit dem Tool *rsnapshot* gesichert werden.

Dieser Bereich wird mit `opsi-backup` **nicht** gesichert.

TFTP-Verzeichnis

Das TFTP-Verzeichnis beinhaltet Konfigurationsdateien für den Bootvorgang per PXE. Diese Verzeichnis befindet sich unter `/tftpboot/` auf den meisten Systemen. Auf openSUSE und SLES ist dieses Verzeichnis `/var/lib/tftpboot/opsi/`.

Möglicherweise angepasste Dateien sind bspw. `linux/pxelinux.cfg/default.menu` bzw. `linux/pxelinux.cfg/default.nomenu`. Diese Dateien werden bei der Installation von opsi-linux-bootimage mit Defaultwerten angelegt. Für ein Disaster Recovery sind diese nicht zwingend nötig.

Dieser Bereich wird mit `opsi-backup` **nicht** gesichert.

5.5.5 Das opsi-backup Programm

Mit dem Kommandozeilenprogramm `opsi-backup` existiert ein Werkzeug, das die Erstellung und das Wiederherstellen einfacher Backups komfortabel erledigt.

Dazu lässt sich `opsi-backup` mit drei grundlegenden Befehlen steuern: `create`, `restore` und `verify`.

Die Option `--help` gibt einen detaillierten Überblick über alle Optionen, die `opsi-backup` akzeptiert.

Ein mit `opsi-backup` erstelltes Backup ist ein Rohbackup, das bedeutet, es werden keine Dateien auf logischer Ebene gesichert, sondern es werden Sicherungen der in den Backends abgelegten Dateien in den entsprechenden Strukturen angefertigt.

Ein solches Backup lässt sich daher auch nur für eine identische Backendkonfiguration zurückspielen.

Ein mit `opsi-backup` erstelltes Backup ist immer ein Vollbackup (`opsi-backup` unterstützt keine inkrementellen oder differenziellen Backups).

Zu beachten ist, dass `opsi-backup` keine Sicherung der [Depot Dateien](#), der [Workbench Dateien](#) sowie der [Repository Dateien](#) durchführt. Diese Dateien sollten daher anderweitig gesichert werden.

Der mit `opsi-backup` erstellte Backup file ist eine komprimierte tar Datei, deren Inhalt sich entsprechend auch anschauen lässt.

```
opsi-backup --help
```



Achtung

Ein Backup, das mit `opsi-backup` erstellt wird, kann unter anderem Passwörter und PC-Keys enthalten, und sollte daher entsprechend sicher archiviert werden.

Ein Backup anlegen

Das Anlegen eines neuen opsi Backups erfolgt mit dem Befehl `opsi-backup create`. Wird dieser Befehl ohne weitere Parameter angegeben erstellt das Programm ein Archiv mit allen Daten der Backends sowie der Konfiguration. Der Dateiname wird dabei automatisch generiert. Für den Befehl `opsi-backup create` sind zusätzliche Programmhilfen verfügbar, welche über die Option `--help` ausgegeben werden.

```
opsi-backup create
opsi-backup create --help
```

Es ist auch möglich, den Dateinamen oder das Zielverzeichnis des neuen Backups vorzugeben. Dazu wird einfach ein Dateiname oder ein Zielverzeichnis einfach an den entsprechenden Befehl angehängt. Wird ein Verzeichnis übergeben, generiert `opsi-backup` automatisch einen Dateinamen in diesem Verzeichnis. Ein durch `opsi-backup` generierter Dateiname hat die Form `<hostname>_<opsi-version>_<datum>_<uhrzeit>` und ist daher gut zur Archivierung mehrerer Backups geeignet. Wird ein Dateiname fest vorgegeben, so wird ein älteres Backup mit dem selben Namen durch `opsi-backup` überschrieben.

```
opsi-backup create /mnt/backup/opsi_backup.tar.bz2
opsi-backup create /mnt/backup/
```

Zusätzlich ermöglicht das `create` Kommando die Steuerung des Backups mittels der folgenden Optionen:

- `--backends {file,mysql,dhcp,all,auto}`

Ermöglicht die Auswahl der Backends, die in dem Backup eingeschlossen werden sollen. Diese Option kann mehrfach angegeben werden, um mehrere Backends anzugeben. Die Option `--backends=all` steht für alle Backends. Die Voreinstellung (default) für diese Optionen ist `--backends=auto`, was dafür sorgt, dass `opsi-backup` versucht, die verwendeten Backends anhand der Konfigurationsdatei `/etc/opsi/backendManager/dispatch.conf` zu ermitteln. Im Moment werden folgende Backends unterstützt: `mysql, file, dhcp`

```
opsi-backup create --backends=file --backends=mysql
opsi-backup create --backends=all
```

Tip

Wenn Sie ein nicht unterstütztes Backend (wie z.B. `ldap`) verwenden, so können Sie vor dem Backup dieses mit dem Befehl `opsi-convert` in ein Backend konvertieren, dass sich per `opsi-backup` sichern lässt.

- `--no-configuration`

Schließt die [Opsi Konfiguration](#) aus dem Backup aus.

```
opsi-backup create --no-configuration
```

- `-c [{gz,bz2,none}]`, `--compression [{gz,bz2,none}]`

Spezifiziert die Kompressionsmethode, mit der das Archiv komprimiert werden soll. `none` steht hier für nicht komprimieren, die Standardkompression (default) ist `bz2`.

```
opsi-backup create -c bz2
```

- `--flush-logs`

Die Sicherung des `mysql`-Backends erfolgt intern über einen `mysqldump` Befehl. Das bedeutet, dass die Daten genauso gesichert werden, wie die Datenbank sie zu diesem Zeitpunkt sieht (unabhängig davon ob die Daten schon auf Platte stehen oder nur im Speicher). Somit ist das erstellte Backup evtl. aktueller und unterscheidet sich vom Stand der Datenbankdateien. Möchte man dies vermeiden, so müssen die von `mysql` im Speicher gehaltenen Daten vorher auf die Festplatte geschrieben werden. Ist die Option `--flush-logs` angegeben, wird `opsi-backup` versuchen, diese Operation durchzuführen (also die Daten aus dem Speicher auf die Platten zuschreiben). Allerdings benötigt der entsprechende Datenbankuser der opsi Datenbank dazu die entsprechende MySQL Berechtigung `RELOAD`. Standardmäßig wird der opsi Benutzer aber ohne dieses Recht angelegt! Besitzt er diese nicht (und die Option `--flush-logs` ist angegeben) wird das Backup fehlschlagen. Verwenden Sie daher diese Option nur, wenn Sie vorher die Rechte des Datenbankusers angepasst haben.

```
opsi-backup create --backends=mysql --flush-logs
```

Beispiel

```
opsi-backup create --no-configuration --backends=all opsi_backup.tar.bz2
```

Backups archivieren

Von Haus aus bringt `opsi-backup` keine Funktionen zum Archivieren von Backups mit. Der Administrator hat daher Sorge zu tragen, dass erzeugte Backups sicher und versioniert abgelegt werden. Außerdem löscht `opsi-backup` niemals selbstständig ältere Backup Version (außer sie werden mittels `create` überschrieben). Da `opsi-backup` immer Vollbackups und keine inkrementellen Backups anlegt, kann es schnell zu großen Datenmengen kommen. Hier muss ebenfalls der Administrator Sorge tragen, dass ältere Backups wenn nötig regelmässig gelöscht werden.

Ein Backup verifizieren

Mit dem Befehl `opsi-backup verify` kann das Archiv auf interne Integrität geprüft werden. Diese Prüfung ist keine logische Prüfung der Daten, es handelt sich um eine reine Prüfung auf die Korrektheit der im Archiv gespeicherten Daten. Für den Befehl `opsi-backup verify` sind zusätzliche Programmhilfen verfügbar, welche über die Option `--help` ausgegeben werden.

Beispiel

```
opsi-backup verify opsi_backup.tar.bz2
opsi-backup verify --help
```

Tip

Wird der Befehl `opsi-backup verify` explizit auf der Konsole aufgerufen ist es häufig sinnvoll, die `opsi-backup` Standardausgabe zu aktivieren: `opsi-backup -v verify opsi_backup.tar.bz2`

Ein Backup wiederherstellen

Das Wiederherstellen des Archivs erfolgt mit dem Befehl `opsi-backup restore`. Dabei werden (per default) die Backends anhand der aktuellen Konfiguration eingespielt. Es kann also kein reines Backend Backup wiederhergestellt werden, ohne dass eine opsi Konfiguration vorhanden ist. Der Befehl `opsi-backup restore` braucht als Parameter das Backup Archiv, aus dem Daten wiederhergestellt werden. Für den Befehl `opsi-backup restore` sind zusätzliche Programmhilfen verfügbar, welche über die Option `--help` ausgegeben werden.

`opsi-backup restore` akzeptiert folgende Optionen:

- `--backends {file,mysql,dhcp,auto,all}`
Stellt das spezifizierte Backend wieder her. Diese Option kann mehrfach angegeben werden, um mehrere Backends anzugeben. Die Option `--backends=all` steht für alle Backends.
Als Voreinstellung (default) wird die Option `--backends=auto` verwendet, was dazu führt, dass `opsi-backup` versucht, anhand der Konfigurationsdatei `/etc/opsi/backendManager/dispatch.conf` festzustellen, welche Backends wiederherzustellen sind.

```
opsi-backup restore --backends=file --backends=mysql opsi_backup.tar.bz2
opsi-backup restore --backends=all opsi_backup.tar.bz2
```



Achtung

Wenn Sie seit der Erstellung des Backups das Backend gewechselt haben, so wird die default Einstellung keine Daten zurück sichern.

- `--configuration`
Stellt die [Opsi Konfiguration](#) wieder her. Diese Option ist beim `restore` Vorgang kein default.

```
opsi-backup restore --configuration opsi_backup.tar.bz2
```

- `-f, --force`
`opsi-backup` führt vor dem Wiederherstellen eines Backups, eine Sicherheitsprüfung durch, um zu überprüfen, ob die aktuelle opsi Installation mit der Installation des Backups übereinstimmt (opsi Version, OS-Version, Host- und Domain Name). Mit dieser Option lässt sich diese Prüfung umgehen.

```
opsi-backup restore -f opsi_backup.tar.bz2
```

Beispiel

```
opsi-backup restore --configuration --backends=all opsi_backup.tar.bz2
```


5.6 Datenhaltung von opsi (Backends)

5.6.1 file-Backend

Bei Verwendung des *file-Backends* liegen die Konfigurationsinformationen in Ini-Dateien auf dem Server.

Wesentliche Merkmale des Backends *file* :

- Aktuelles Defaultbackend von opsi
- Die Dateien dieses Backends liegen unter `/var/lib/opsi`.
- Ist implementiert in der Annahme, dass der FQDN des Servers auf welchem das Backend verwendet wird dem FQDN des `:configserver:` in opsi entspricht.

Inhalt und Aufbau dieser Dateien ist im Kapitel Abschnitt [5.7.4](#) näher erläutert.

5.6.2 mysql-Backend

mysql-Backend für Inventarisierungsdaten (Übersicht und Datenstruktur)

Die Daten der Hardware- und Softwareinventarisierung werden per default über das opsi *file-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* :

- Optional (nicht das default Backend)
- Für Inventarisierungsdaten kostenfrei, für die Nutzung für sonstige Daten benötigen Sie eine kostenpflichtige Freischaltung.
- Fein granulierte Datenstruktur zur Datenhaltung und zusätzlich vereinfachtes Datenmodell für Abfragen.
- Eine Historyfunktion, welche Änderungen an den Inventarisierungsdaten protokolliert.

Bedingt durch die sehr unterschiedliche Natur der zu inventarisierenden Hardwarekomponenten 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
 - *HARDWARE_CONFIG*... beschreibt Konfiguration der konkreten Netzwerkkarte z.B. MAC-Adresse. Die beiden Tabellen sind über das Feld *hardware_id* miteinander verbunden.

Ähnlich sieht es für die Softwareinventarisierung aus. Auch hier beschreibt die Tabelle *Software* die insgesamt gefundene Software während die Tabelle *Software_Config* die Client spezifische Konfiguration speichert.

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
HOST
SOFTWARE
SOFTWARE_CONFIG

```

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 = Seriennummer
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: Liste aller Festplatten:

```

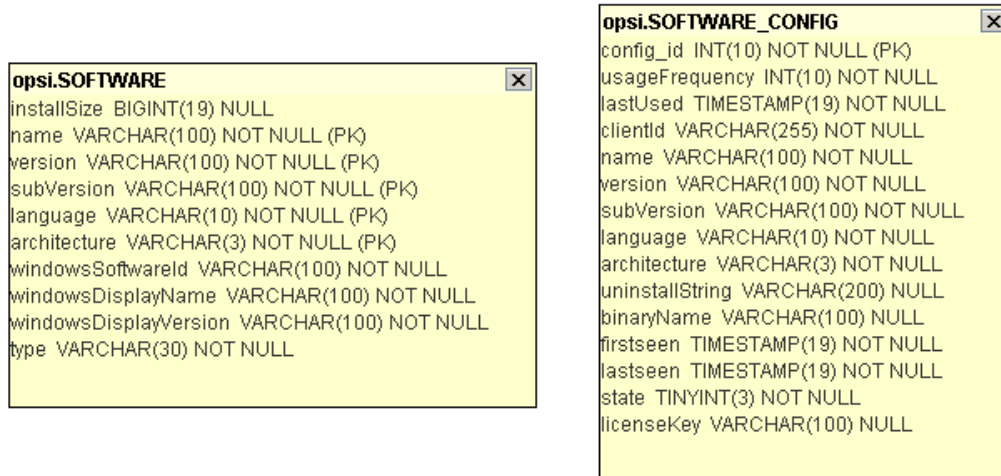
SELECT * FROM HARDWARE_DEVICE_HARDDISK_DRIVE D
LEFT OUTER JOIN HARDWARE_CONFIG_HARDDISK_DRIVE H ON D.hardware_id=H.hardware_id ;

```

Die Softwareinventarisierung verwendet als Hauptschlüssel die folgenden Felder:

- Name
Dieser ist der *windowsDisplayName* bzw. wenn dieser nicht vorhanden ist die *windowsSoftwareId*. Beide werden aus der Registry ermittelt:
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall bzw.
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\<id> DisplayName
- Version
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\<id> DisplayVersion
- SubVersion
- Language
- Architecture (32 Bit / 64 Bit)

In der Tabelle *Software_config* sind diese Felder zum Feld *config_id* zusammengefasst.



The image shows two overlapping windows displaying database schemas. The left window is titled 'opsi.SOFTWARE' and lists the following fields: installSize (BIGINT(19) NULL), name (VARCHAR(100) NOT NULL (PK)), version (VARCHAR(100) NOT NULL (PK)), subVersion (VARCHAR(100) NOT NULL (PK)), language (VARCHAR(10) NOT NULL (PK)), architecture (VARCHAR(3) NOT NULL (PK)), windowsSoftwareId (VARCHAR(100) NOT NULL), windowsDisplayName (VARCHAR(100) NOT NULL), windowsDisplayVersion (VARCHAR(100) NOT NULL), and type (VARCHAR(30) NOT NULL). The right window is titled 'opsi.SOFTWARE_CONFIG' and lists: config_id (INT(10) NOT NULL (PK)), usageFrequency (INT(10) NOT NULL), lastUsed (TIMESTAMP(19) NOT NULL), clientId (VARCHAR(255) NOT NULL), name (VARCHAR(100) NOT NULL), version (VARCHAR(100) NOT NULL), subVersion (VARCHAR(100) NOT NULL), language (VARCHAR(10) NOT NULL), architecture (VARCHAR(3) NOT NULL), uninstallString (VARCHAR(200) NULL), binaryName (VARCHAR(100) NULL), firstseen (TIMESTAMP(19) NOT NULL), lastseen (TIMESTAMP(19) NOT NULL), state (TINYINT(3) NOT NULL), and licenseKey (VARCHAR(100) NULL).

opsi.SOFTWARE	opsi.SOFTWARE_CONFIG
installSize BIGINT(19) NULL	config_id INT(10) NOT NULL (PK)
name VARCHAR(100) NOT NULL (PK)	usageFrequency INT(10) NOT NULL
version VARCHAR(100) NOT NULL (PK)	lastUsed TIMESTAMP(19) NOT NULL
subVersion VARCHAR(100) NOT NULL (PK)	clientId VARCHAR(255) NOT NULL
language VARCHAR(10) NOT NULL (PK)	name VARCHAR(100) NOT NULL
architecture VARCHAR(3) NOT NULL (PK)	version VARCHAR(100) NOT NULL
windowsSoftwareId VARCHAR(100) NOT NULL	subVersion VARCHAR(100) NOT NULL
windowsDisplayName VARCHAR(100) NOT NULL	language VARCHAR(10) NOT NULL
windowsDisplayVersion VARCHAR(100) NOT NULL	architecture VARCHAR(3) NOT NULL
type VARCHAR(30) NOT NULL	uninstallString VARCHAR(200) NULL
	binaryName VARCHAR(100) NULL
	firstseen TIMESTAMP(19) NOT NULL
	lastseen TIMESTAMP(19) NOT NULL
	state TINYINT(3) NOT NULL
	licenseKey VARCHAR(100) NULL

Abbildung 53: Datenbankschema: Softwareinventarisierung

mysql-Backend für Konfigurationsdaten (Übersicht)

Das *mysql-Backend* für Konfigurationsdaten steht seit opsi 4.0 zur Verfügung.

Dieses Modul ist momentan eine kofinanzierte opsi Erweiterung. Das bedeutet die Verwendung ist nicht kostenlos. Weitere Details hierzu finden Sie in Abschnitt [9.1](#).

Das *mysql-Backend* hat den Vorteil der höheren Performanz insbesondere bei großen Installationen.

Hier eine Übersicht über die Datenstruktur:

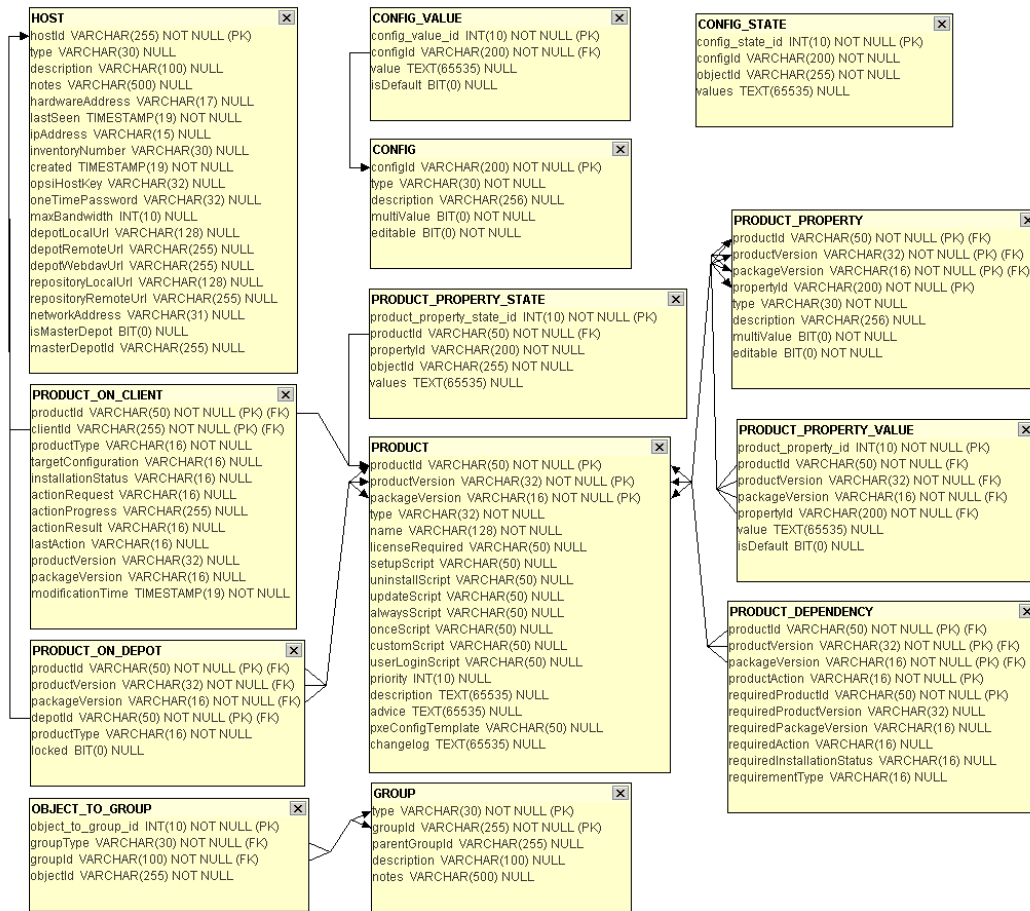


Abbildung 54: Datenbankschema: Konfigurationsdaten

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
```

Achtung

MySQL-Server verwendet seit Version 5.7 den vorher optionalen *strict mode* nun standardmäßig. Dies führt zu einem Fehlschlag des Befehls `opsi-setup --configure-mysql`. Dementsprechend sollte vor dem Befehlsaufruf die Datei `/etc/mysql/mysql.conf.d/mysqld.cnf` editiert werden.

In der `[mysqld]` Sektion muss nun folgende Zeile eingefügt werden:

```
sql_mode=NO_ENGINE_SUBSTITUTION
```

Danach muss der Dienst mysql neu gestartet werden: `service mysql restart`

Es ist nun möglich fort zu fahren.

Mit dem Befehl `opsi-setup --configure-mysql` kann nun die Datenbank aufgebaut werden.

Eine Beispiel-Sitzung:

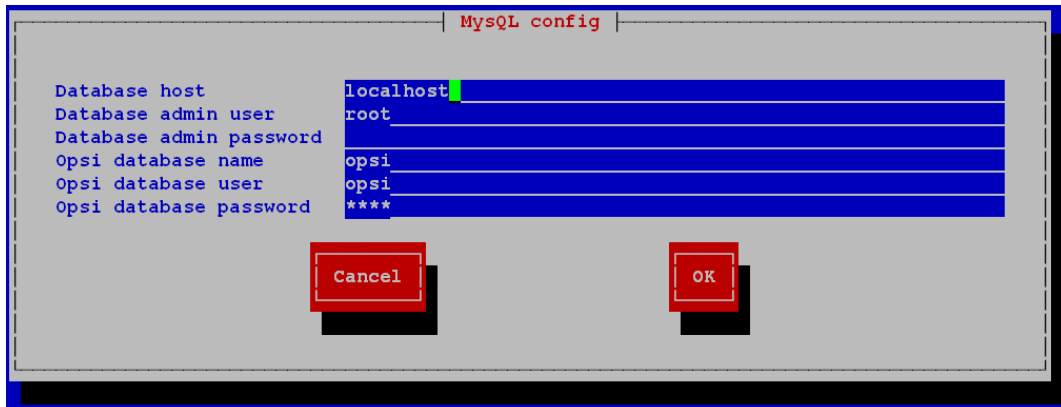


Abbildung 55: opsi-setup --configure-mysql: Eingabemaske

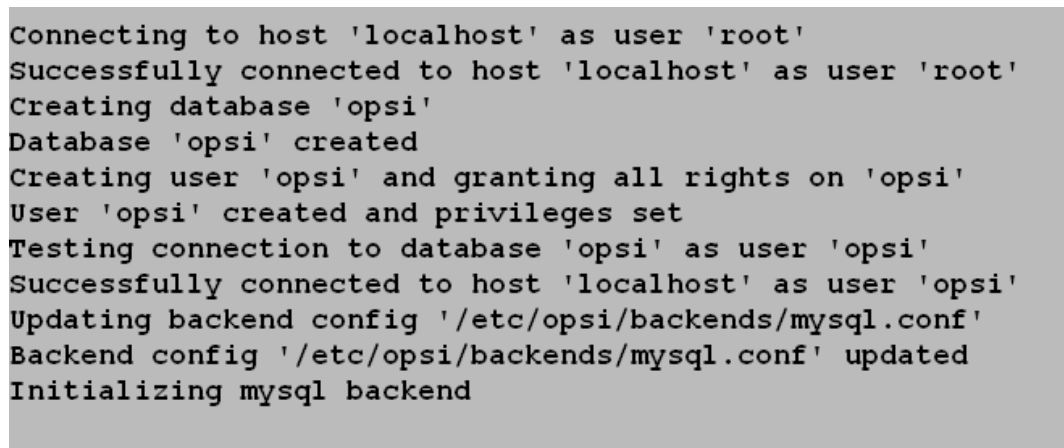


Abbildung 56: opsi-setup --configure-mysql: Ausgabe

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

Als nächstes muss in der `/etc/opsi/backendManager/dispatch.conf` eingetragen werden, dass das *mysql-Backend* auch verwendet werden soll. Eine genaue Beschreibung zu dieser Konfiguration finden Sie im Kapitel *Backend-Konfiguration* des *getting-started Handbuchs*. Die Datei selbst enthält eine Reihe von Beispielen typischer Konfigurationen. Eine Konfiguration für *mysql-Backend* (ohne internen DHCPD) sieht so aus:

```

backend_.*      : mysql, opsipxeconfd
host_.*        : mysql, opsipxeconfd
productOnClient_.* : mysql, opsipxeconfd
configState_.* : mysql, opsipxeconfd
.*            : mysql

```

Nach Abschluss dieser Konfigurationsarbeit müssen Sie den folgenden Befehlen die Benutzung der jetzt konfigurierten und konvertierten Backend aktivieren:

```

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

```



Achtung

Der Dienst *opsiconfd* hat per Default keine Abhängigkeit zu MySQL, da opsi einerseits ohne MySQL-Backend verwendet, andererseits der Dienst auch auf einem anderen Server starten kann. Außerdem unterstützt nicht jedes Init-System die Formulierung von Abhängigkeiten. Bitte entnehmen Sie der Dokumentation Ihres Betriebssystems wie eine solche Konfiguration gemacht wird.

Konfigurieren der MySQL-Datenbank zum Zugriff von außen

Die vorliegende Datenbank muss so konfiguriert werden, dass ein Zugriff von außen möglich ist, also nicht nur Verbindungen von *localhost* akzeptiert werden.

Bitte informieren Sie sich um Handbuch der von Ihnen verwendeten Datenbank über die nötigen Schritte.

5.6.3 HostControl-Backend

Das HostControl-Backend speichert keine Konfigurationsdaten, sondern dient der Steuerung von opsi-Clients. Hierzu gehören beispielsweise das Starten von Clients per Wake-On-LAN oder das Senden von Steuerungsbefehlen an den opsi-client-agent.

Die Konfiguration des HostControl-Backends wird in der Konfigurationsdatei */etc/opsi/backends/hostcontrol.conf* vorgenommen. Konfigurations-Optionen sind hierbei:

- **opsiclientdPort:**
Netzwerk-Port für die Verbindungsaufnahme zu einem opsi-client-agent.
- **hostRpcTimeout:**
Timeout (in Sekunden) bei der Verbindungsaufnahme zu einem opsi-client-agent.
- **resolveHostAddress:**
Steht diese Option auf **True**, wird bei einem Verbindungsaufbau vom opsi-server zu einem opsi-client die IP-Adresse des Clients bevorzugt über die Namensauflösung ermittelt. Um die im Backend von opsi hinterlegte IP-Adresse zu bevorzugen ist die Option auf **False** zu setzen.
- **maxConnections:**
Maximale Anzahl simultaner Verbindungen zu opsi-client-agents.
- **broadcastAddresses:**
Liste von Broadcast-Adressen für das Versenden von Wake-On-LAN-Broadcasts.

5.6.4 HostControlSafe-Backend

Eine Besonderheit beim Standardverhalten von opsi4.0 Methoden ist, dass bei einer Abfrage ohne Angaben von Parametern, alle Objekte abgerufen werden. Beispielsweise gibt der Befehl "host_getObjects" ohne Parameter aufgerufen, alle Host-Objekte zurück. Dieses Verhalten ist im HostControl-Backend etwas problematisch. Besonders bei den beiden Befehlen: *hostControl_shutdown* und *hostControl_reboot*. In diesen Fällen würde ein Aufruf dieser Methoden ohne Parameter alle Clients herunterfahren bzw. neustarten.

Deshalb gibt es mit Service Release opsi 4.0.3 an dieser Stelle zwei Änderungen:

- Die Methoden: *hostControl_shutdown* und *hostControl_reboot* brechen seit dieser Release mit dem opsi 4.0 Standardverhalten. Diese beiden Methoden geben nun eine Fehlermeldung zurück, wenn kein Parameter übergeben wurde.
- Es wurde ein neues Backend (**HostControlSafe-Backend**) eingeführt, welches Standardmäßig bei allen Methoden eine Fehlermeldung ausgegeben wird, wenn keine korrekte Angaben zu den Clients übergeben wird. Um mit einer Methode vom HostControlSafe-Backend alle Clients an zu sprechen, kann man das *-Zeichen verwenden:


```
opsi-admin -d method hostControlSafe_shutdown *
```

Aus den oben genannten Gründen, empfehlen wir hostControlSafe-Methoden zu verwenden, wenn man etwas unsicher auf der Konsole ist oder neu anfängt sich mit den Servicemethoden zu beschäftigen.

5.6.5 Konvertierung zwischen Backends

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

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

```
opsi-convert --help

Usage: opsi-convert [options] <from> <to>
Convert an opsi database into an other.
Options:
  -h          show this help text
  -V          show version information
  -q          do not show progress
  -v          increase verbosity (can be used multiple times)
  -c          clean destination database before writing
  -s          use destination host as new server
  -l <file>  log to this file

<from> and <to> can be:
  - the name of a backend as defined in /etc/opsi/backends (file, ldap, ...)
  - the url of a opsi configuration service
    http(s)://<user>@<host>:<port>/rpc
```

5.6.6 Bootdateien

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

5.6.7 Absicherung der Shares über verschlüsselte Passwörter

Der *opsi-client-agent* greift auf die vom *opsi-server* zur Verfügung gestellten Shares zu, um die dort liegende Software zu installieren 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-client-agent* ein Zugriff auf die Authentifizierungsdaten zu ermöglichen, wird für jeden Client bei seiner Erzeugung in opsi ein spezifischer Schlüssel (*opsi-host-Schlüssel*) erzeugt. Dieser Schlüssel wird zum einen (beim *file-Backend*) in der Datei `/etc/opsi/pkeys` abgelegt und zum anderen dem PC bei der Reinstallation übergeben. Der übergebene Schlüssel wird im Rahmen der Installation des *opsi-client-agent* in der Datei `c:\program files\opsi.org\opsi-client-agent\opsiclientd\opsiclientd.conf` so abgelegt, dass nur Administratoren Zugriff darauf haben. Ebenso hat auf dem *opsi-server* nur root und Mitglieder der Gruppe *opsiadmin* Zugriff auf die Datei `/etc/opsi/pkeys`. 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 jeder Aktivierung des *opsi-client-agent* neu gelesen, so dass eine Änderung des *pcpatch* Passwortes jederzeit möglich ist und der Client auf verschlüsseltem Wege das veränderte Passwort erfährt.

5.7 Wichtige Dateien des opsi-servers

5.7.1 Allgemeine Konfigurationsdateien in /etc

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

Beispiel:

```
192.168.2.106 dplaptop.uib.local dplaptop # this opsi-server
192.168.2.153 schleppi.uib.local
192.168.2.178 test_pc1.uib.local # Test-PC PXE-bootprom
```

Die Ausgabe von:

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

Das Ergebnis sollte beispielsweise so aussehen:

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

Sieht das Ergebnis nicht so aus (enthält z.B. *127.0.0.1* oder *localhost*), dann müssen Sie Ihre `/etc/hosts` oder Namensauflösung zunächst korrigieren.

`/etc/group`

Hier müssen zwei Gruppen angelegt sein: *pcpatch* und *opsiadmin*. In der Gruppe *pcpatch* sollten alle Benutzer sein, die mit Paketverwaltung zu tun haben. In der Gruppe *opsiadmin* müssen alle user sein, die den opsi-confd-Webservice verwenden wollen z.B. über den opsi-configed.

`/etc/opsi/backends/`

Konfigurationsdateien der verwendeten Backends.

`/etc/opsi/backendManager/`

- `acl.conf`
Konfiguration der Zugriffsrechte auf die opsi Methoden. Hierbei können für die Basis-Methoden des Webservices Zugriffsrechte für bestimmte Benutzer und auf bestimmte Attribute eingeschränkt werden.

- **dispatch.conf**
Konfiguration welche der unter `/etc/opsi/backends/` konfigurierten Backends wofür verwendet werden sollen.
- **extend.d/**
Verzeichnis der Backenderweiterungen. So liegen hier z.B. die Skripte, welche die opsi 3 Methoden auf die opsi 4 Methoden mappen.

/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.

/etc/opsi/opsi.conf

Seit Version 4.0.2-2

Allgemeine opsi Konfigurationen.

Beispiel:

```
[groups]
fileadminingroup = pcpatch
```

Hintergrund: Die klassischen Installationsvariante mit dem Benutzer: `pcpatch` mit der primären Gruppe: `pcpatch` funktioniert nicht mit Samba 4. Da Samba4 den grundlegenden Restriktionen von Active-Directory unterliegt, sind Gruppen mit der gleichen Bezeichnung wie User (wie in Unix/Linux üblich) nicht mehr erlaubt. Aus diesem Grund wurde eine neue Konfigurationsdatei eingeführt: `/etc/opsi/opsi.conf`, über die gesteuert wird, wie die Gruppe für den Samba-Zugriff auf die Freigaben bestimmt wird. So wird bei Distributionen mit Samba 4 nun über diese Datei der Gruppenname `pcpatch` umbenannt und heißt von nun an: `opsifileadmins`. Das bedeutet, dass die User, die Zugriffsrechte für die Freigaben von opsi erhalten müssen (opsi-Paketierer) nicht Mitglied der Gruppe `pcpatch` werden können, sondern Mitglied der Gruppe `opsifileadmins` sein müssen.

/etc/opsi/modules

Ab Version 3.4

Von der uib gmbh signierte Datei zur Freischaltung kostenpflichtiger Features. Wird die Datei verändert, verliert sie Ihre Gültigkeit. Ohne diese Datei stehen nur die kostenlosen Features zur Verfügung.

/etc/opsi/opsiconfd.conf

Ab Version 3

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

/etc/opsi/opsiconfd.pem

Ab Version 3

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

/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.

/etc/opsi/opsi-product-updater.conf

Konfigurationsdatei für den opsi-product-updater. Siehe Abschnitt 5.3.3

/etc/opsi/version

Enthält die Versionsnummer des installierten opsi.

/etc/init.d/

Start-Stop Skripte für: * opsi-atftpd * opsiconfd * opsipxeconfd

5.7.2 Bootdateien

Bootdateien in /tftpboot/linux

- **pxelinux.0**
Bootfile, der im ersten Schritt vom PXE-Bootprom geladen wird.
- **install** und **miniroot.gz**
Installationsbootimage, das per tftp an den Client bei der Reinstallation übertragen wird.

Bootdateien in /tftpboot/linux/pxelinux.cfg

- **01-<mac adresse>** bzw. **<IP-NUMMER-in-Hex>**
Dateien mit der Hardwareadresse des Clients und dem Prefix 01 - sind auf dem *opsi-server* als clientspezifische Bootfiles zu finden. Sie sind zumeist über den *opsipxeconfd* als named pipes erzeugt und sollen eine Reinstallation des Clients einleiten.
- **default**
Die Datei default wird geladen, wenn es keine clientspezifischen Dateien gibt. Wird diese Datei geladen, so bootet der Client danach lokal weiter.
- **install**
Informationen zum boot des Installationsbootimages, die vom *opsipxeconfd* in die named pipe geschrieben werden.

5.7.3 Dateien in /var/lib/opsi

/var/lib/opsi/repository

Hier werden *Produkt-Pakete* gespeichert, welche über den Aufruf des *opsi-product-updater* auf den Server geladen werden.

Weiterhin werden hier *Produkt-Pakete* gespeichert, welche über den Aufruf des *opsi-package-manager* installiert werden, wenn dieser mit der Option **-d** aufgerufen wird.

/var/lib/opsi/depot

Dieses Verzeichnis ist (read-only) als Samba share *opsi_depot* freigegeben. Bei alten opsi-Installationen war dieses Verzeichnis */opt/pcbin/install*. Sollte dieses Verzeichnis noch existieren, so ist es durch einen Symlink mit */var/lib/opsi/depot* verbunden.

/var/lib/opsi/ntfs-images

In diesem Verzeichnis werden (per default) Partitionsimages abgelegt, welche mit dem Netboot-Produkt *opsi-clonezilla* ausgelesen werden.

Weitere Verzeichnisse

Die restlichen Verzeichnisse in `/var/lib/opsi` (`config` und `audit`) sind Verzeichnisse des *file-Backends*, welche im folgenden Kapitel beschrieben sind.

5.7.4 Dateien des file Backends

`/etc/opsi/pckey`s

Hier sind die clientspezifischen *opsi-host-Schlüssels* sowie der Schlüssel des Servers selber abgelegt.

Beispiel:

```
schleppi.uib.local:fdc2493ace4b372fd39dbba3fcd62182
laptop.uib.local:c397c280fc2d3db81d39b4a4329b5f65
pcbon13.uib.local:61149ef590469f765a1be6cfbacbf491
```

`/etc/opsi/passwd`

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

Übersicht `/var/lib/opsi`

Die Dateien des file Backends von opsi 4 finden sich standardmäßig in `/var/lib/opsi/config/`. Das folgende Schema gibt einen Überblick der Verzeichnisstruktur:

```
/var/lib/opsi-|
              |-depot                opsi_depot share
              |-repository            opsi package repository used by opsi-product-updater opsi-package-\  
manager      |-audit                inventory - files
              !-config/-|            config share
                  |-clientgroups.ini  client groups
                  |-config.ini        Host Parameter (Global Defaults)
                  |-clients/          <pcname.ini> files
                  |-products/        product control files
                  !-depots            depot description files

+audit/
  global.<Type> (Allgemeine Hard-, bzw. Softwareinformationen)
  <FQDN>.<Type> (Hard-, bzw. Softwareinformationen der Clients)

clientgroups.ini (enthält HostGroups)

+clients/
  <FQDN>.ini (Informationen der Clients)
config.ini (enthält Configs)

+depots/
  <FQDN>.ini (Informationen der Server)

+products/
  <ID>_<ProdVer>-<PackVer>.<Type> (Informationen der Products)

+templates/
  pcproto.ini (Vorlage für Clients)
  <FQDN>.ini (Vorlage für spezifische Clients)
```



Warnung

Vom Editieren der Dateien wird dringend abgeraten!

Konfigurationsdateien im Detail

In den folgenden Kapiten werden die Konfigurationsdateien des file-Backends im Detail vorgestellt.

./clientgroups.ini

Die Datei enthält die Informationen über Client-Gruppen.

```
[<GroupId>]
<HostId> = 1 #aktiv
<HostId> = 0 #inaktiv
```

./config.ini

Hier finden sich die Defaultwerte der Serverkonfiguration wie im *opsi-configed* im Tab *Host Parameter* angezeigt.

./clients/<FQDN>.ini

In der dieser Datei werden die Client spezifischen Konfigurationen zusammen gefasst. Die Informationen werden mit denen aus der *<depot-id>.ini* zusammengefasst, wobei Informationen aus der *<FQDN>.ini* Vorrang haben.

Diese Dateien sind folgendermaßen aufgebaut:

Die Sektion *info* enthält alle direkt auf den Client bezogene Informationen, wie z.B. die Beschreibung.

```
[info]
description = <String>
created = <Date> #format: 'YYYY-MM-DD HH:MM:SS'
lastseen = <Date> #format: 'YYYY-MM-DD HH:MM:SS'
inventorynumber = <String>
notes = <String>
hardwareaddress = <MAC> #format: 'hh:hh:hh:hh:hh:hh'
ipaddress = <IP> #format: 'nnn.nnn.nnn.nnn'
onetimepassword = <String>
```

Die folgende Sektion beschreibt die aktuellen Zustände der Produkte auf dem Client. Wenn keine Einträge vorhanden sind, wird *not_installed:none* angenommen.

```
[<Type>_product_states] #'Local-', bzw. 'NetbootProduct'
<ProductId> = <InstallationStatus>:<ActionRequest>
```

Genauere Informationen stehen dazu in den, zu den jeweiligen Produkten zugehörigen, Sektionen.

```
[<ProductId>-state]
producttype = <Type> #'Local-', bzw. 'NetbootProduct'
actionprogress = <String>
productversion = <ProdVer>
packageversion = <PackVer>
modificationtime = <Date> #format: 'YYYY-MM-DD HH:MM:SS'
lastaction = <ActionRequest>
actionresult = <ActionResult>
targetconfiguration = <InstallationStatus>
```

/var/lib/opsi/config/templates

Hier findet sich die Datei *pcproto.ini*, welche das Standardtemplate zur Erzeugung neuer Client-Ini-Dateien ist und besitzt dieselbe Struktur. Wenn bestimmte Clients abweichende Informationen erhalten sollen, kann man auch jeweils eine *<FQDN>.ini* in diesem Verzeichnis ablegen.

/var/lib/opsi/config/depots/

Hier findet sich die Dateien der *opsi-depotserver*, die ebenfalls mit `<depot-id>.ini` gespeichert werden. Hier wird u.a. die Erreichbarkeit des Depots abgelegt.

```
[depotshare]
remoteurl = smb://<NetBiosName>/<Path>
localurl = file://<Path>

[depotserver]
notes = <String>
network = <IP>
description = <String>
hardwareaddress = <MAC>
ipaddress = <IP>
inventorynumber = <String>

[repository]
remoteurl = webdavs://<FQDN>:<Port>/<Path>
localurl = file://<Path>
maxbandwidth = <Integer> #in Bytes
```

Hier finden sich aber auch die Informationen, welche opsi-Produkte, in welcher Version und mit welchen Property Defaultwerten, auf dem Depot installiert sind.

Product control files in /var/lib/opsi/config/products/

Die product control files 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 control files 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 Produktabhängigkeiten.

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 Verschlüsselungs 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 *opsi-winst*-Skript 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.
- [Product]-*name*
ist der Klartextname des Produkts.
- [Product]-*Description*
ist eine ergänzende Beschreibung zum Produkt, die z.B. im *opsi-configed* unter *Beschreibung* angezeigt wird.
- [Product]-*Advice*
ist eine ergänzende Beschreibung (in der Regel) zum Umgang mit dem Produkt, die zu beachten ist und im *opsi-configed* unter *Notiz* angezeigt wird.
- [Product]-*version*
ist die Version der eingepackten Software.
- [Product]-*Priority*
beeinflusst zusammen mit den Produktabhängigkeiten die Installationsreihenfolge.
- [Product]-*productClasses*
wird zur Zeit noch nicht verwendet (und auch nicht angezeigt).
- [ProductProperty]-*type*
Typ des properties: (unicode/boolean)
- [ProductProperty]-*name*:
Anzeigename der Eigenschaft.
- [ProductProperty]- *multivalue*
Kann dieses Property eine Liste von Werten enthalten (True/False)

- [ProductProperty]- *editable*
Kann dieses Property frei editiert werden (oder kann nur aus einer vorgegebenen Liste ausgewählt werden). (True/False)
- [ProductProperty]-*description*:
Beschreibung der Eigenschaft (Tooltip im *opsi-configed*).
- [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, uninstall ...).
- [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, uninstall, 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.

Inventarisierungsdateien /var/lib/opsi/audit

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

5.7.5 opsi Programme und Libraries

Programme in /usr/bin

- `opsipxeconfd`
opsi Daemon, welcher für den PXE-Start der Clients die notwendigen Dateien im tftp-Bereich des Servers verwaltet.
- `opsi-admin`
Kommandozeilen-Interface zur opsi python Library.
- `opsiconfd`
opsi Daemon zur Bereitstellung der opsi Methoden als Webservice und vieles mehr.
- `opsiconfd-guard`
opsi Daemon, der überwacht, ob der *opsiconfd* läuft und diesen im Zweifelsfall neu startet.
- `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-package-manager
Skript zum Installieren und Deinstallieren von opsi-Paketen auf einem opsi-server.
- opsi-setup
Programm für diverse Basiskonfigurationen.

5.7.6 opsi-Logdateien

Die opsi Logdateien haben das Format:

```
[Loglevel] Timestamp Meldung
Die Loglevel sind dabei:
0 = nothing      (absolute nothing)
1 = essential   ("we always need to know")
2 = critical     (unexpected errors that my cause a program abort)
3 = error        (Errors that don't will abort the running program)
4 = warning     (you should have a look at this)
5 = notice      (Important statements to the program flow)
6 = info        (Additional Infos)
7 = debug       (important debug messages)
8 = debug2      (a lot more debug informations and data)
9 = confidential (passwords and other security relevant data)
```

/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. Um in einem solchen Fall an die Logdatei vom bootimage zu kommen, gibt es zwei Wege:

1. Netzwerk geht
Dann kann man per SCP z.B. von Windows aus per WinSCP die Datei /tmp/log holen.
2. Netzwerk geht nicht

Dann hilft der USB-Stick:

- Als root mit pass *linux123* einloggen
- USB-Stick einstecken und ein paar Sekunden warten
- mit `sfdisk -l` prüfen, auf welchem Device der Stick liegt
- mounten
- kopieren
- unmounten

Das Ganze sieht als Beispiel etwa so aus:

```
#sfdisk -l
Disk /dev/sda: 30401 cylinders, 255 heads, 63 sectors/track
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0

   Device Boot Start      End  #cyls  #blocks  Id System
/dev/sda1  *        0+    30401- 30402- 244197528+  7  HPFS/NTFS
/dev/sda2            0      -      0        0  0  Empty
/dev/sda3            0      -      0        0  0  Empty
```

```

/dev/sda4      0      -      0      0      0      0      Empty

Disk /dev/sdb: 1017 cylinders, 33 heads, 61 sectors/track
Units = cylinders of 1030656 bytes, blocks of 1024 bytes, counting from 0

   Device Boot Start      End  #cyls   #blocks   Id System
/dev/sdb1      0+    1016   1017-   1023580    b  W95 FAT32
/dev/sdb2      0      -      0        0      0  Empty
/dev/sdb3      0      -      0        0      0  Empty
/dev/sdb4      0      -      0        0      0  Empty
# mount /dev/sdb1 /mnt
# cp /tmp/log /mnt
#umount /mnt

```

/var/log/opsi/clientconnect

Hier findet sich die Logdatei der auf dem Client laufenden *opsi-client-agent*. Dies ist auf dem client die C:\opsi.org\log\opsiclientd.log.

/var/log/opsi/instlog

Hier findet sich die Logdatei der auf den Clients ausgeführten *opsi-winst*-Skripte. Die Originale liegen auf dem Client unter C:\opsi.org\log\opsiscript.log

/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* eingestellt, zu diesen symbolische Links als <IP-Name>.log erzeugt.

/var/log/opsi/opsipxeconfd.log

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

/var/log/opsi/package.log

Logdatei des opsi-package-manager.

/var/log/opsi/opsi-product-updater.log

Logdatei des opsi-product-updater.

tftp log in /var/log/syslog

Die Logeinträge des tftpd finden sich in */var/log/syslog*.

Damit diese auch aussagerkräftig sind muss der Loglevel des tftpd auf 7 erhöht werden:

In der Datei */etc/inetd.conf* in der Zeile die mit *tftpd* anfängt den Parameter *verbose* auf 7 setzen:

```

tftp  dgram  udp    wait   nobody /usr/sbin/tcpd /usr/sbin/in.tftpd --tftpd-timeout 300 --retry-timeout 5  --\
      mcast-port 1758 --mcast-addr 239.239.239.0-255 --mcast-ttl 1 --maxthread 100 --verbose=7 /tftpboot

```

danach ausführen:

```

killall tftpd
killall -1 inetd

```

c:\opsi.org\log\opsi_loginblocker.log

Logdatei des Loginblockers.

c:\opsi.org\log\opsiclientd.log

Logdatei des opsiclientd.

Wird bei Beendigung auf den Server nach `/var/log/opsi/clientconnect/<pc-ipnummer.log>` kopiert.

c:\opsi.org\log\opsi-script.log

Logdatei des *opsi-winst*.

Wird bei Beendigung auf den Server nach `/var/log/opsi/instlog/<pc-ipnummer.log>` kopiert.

5.8 Upgrade Anleitungen für den opsi-server

Diese finden Sie ab opsi 4.0 in den versionspezifischen *releasenotes* Handbüchern.

5.9 Hinweise zur Dateistruktur des opsi-linux-bootimages unter UCS 4.X

Im Vergleich zu Ubuntu oder Debian liegen die Dateien des opsi-linux-bootimage in einem anderen Verzeichnis, nämlich:

```
/var/lib/univention-client-boot
```

In früheren Versionen des opsi-linux-bootimage wurden Dateien aus dem Verzeichnis `/tftpboot/linux` in das Verzeichnis `/var/lib/univention-client-boot` gelinkt. Dies wurde in mit der Einführung von opsi 4.1 geändert. Im opsi-linux-bootimage Paket von opsi 4.1 wurden die Dateien direkt in das Verzeichnis `/var/lib/univention-client-boot` installiert. Nachdem der opsi-atftpd unter opsi 4.0 Dateien mit einer Größe von mehr als 90MiB erlaubte, wurde dieses opsi-linux-bootimage auch für opsi 4.0 zur Verfügung gestellt. Die Dateistruktur sieht wie folgt aus:

```
ls -l /var/lib/univention-client-boot/
insgesamt 224424
-rw-rw-r-- 1 997 OPSI Depot Servers 12372 Aug 13 13:13 chain.c32
lrwxrwxrwx 1 997 OPSI Depot Servers 15 Aug 13 13:13 install -> vmlinuz-4.17.13
lrwxrwxrwx 1 997 OPSI Depot Servers 11 Aug 13 13:13 install64 -> install-x64
lrwxrwxrwx 1 997 OPSI Depot Servers 19 Aug 13 13:13 install-x64 -> vmlinuz-x64-4.17.13
-rw-rw-r-- 1 997 OPSI Depot Servers 52272 Aug 13 13:13 menu.c32
-rw-rw-r-- 1 997 OPSI Depot Servers 105388996 Aug 13 13:13 miniroot-20180813.bz2
lrwxrwxrwx 1 997 OPSI Depot Servers 21 Aug 13 13:13 miniroot.bz2 -> miniroot-20180813.bz2
-rw-rw-r-- 1 997 OPSI Depot Servers 108394052 Aug 13 13:13 miniroot-x64-20180813.bz2
lrwxrwxrwx 1 997 OPSI Depot Servers 25 Aug 13 13:13 miniroot-x64.bz2 -> miniroot-x64-20180813.bz2
-rw-rw-r-- 1 997 OPSI Depot Servers 15710 Aug 13 13:13 pxelinux.0
drwxrwxr-x 2 997 OPSI Depot Servers 4096 Aug 28 19:19 pxelinux.cfg
-rw-rw-r-- 1 997 OPSI Depot Servers 7763664 Aug 13 13:13 vmlinuz-4.17.13
-rw-rw-r-- 1 997 OPSI Depot Servers 8166656 Aug 13 13:13 vmlinuz-x64-4.17.13
```

Wird der Inetd-Dienst verwendet um den opsi-atftpd zu steuern muss die tftp Zeile in der Datei `/etc/inetd.conf` wie folgt aussehen:

```
tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd --tftp-timeout 300 --retry-timeout 5 --verbose=5 /var/lib/
/univention-client-boot
```

Falls der Inetd-Dienst nicht verwendet wird, sondern der Dienst des opsi-atftpd direkt, muss die Konfiguration unter `/etc/default/atftpd` diesen Inhalt besitzen:

```
USE_INETD=false
OPTIONS="--daemon --port 69 --tftpd-timeout 300 --retry-timeout 5
--mcast-port 1758 --mcast-addr 239.239.239.0-255 --mcast-ttl 1 --maxthread 100 --verbose=5
/var/lib/univention-client-boot "
```

Unter opsi 4.1 kommt der `opsi-tftpd-hpa` zum Einsatz, dieser Patcht die entsprechende Konfigurationsdatei bei der Installation und setzt das `TFTPROOT` Verzeichnis auf `/var/lib/univention-client-boot`

UCS 4.X benötigt noch eine DHCP Richtlinie um einen PXE-Boot von einem UCS System aus zu ermöglichen. Diese Richtlinie muss in den Richtlinieneinstellungen der Domäne vorgenommen werden und betrifft den DHCP Boot. Als Boot-Server trägt man den Server ein, auf welchem die Boot Datei liegt. Der Boot-Dateiname lautet `pxelinux.0`. Da diese Datei direkt im Verzeichnis `/var/lib/univention-client-boot` liegt, muss kein Anführendes Verzeichnis mit angegeben werden.

6 opsi-client

6.1 opsi-client-agent

6.1.1 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 einen Agenten auf dem Client realisiert:

Auf dem Client wird der sogenannte *opsi-client-agent* installiert. Dieser überprüft üblicherweise beim Start des Clients und vor dem Login des Anwenders, anhand von Konfigurations-Informationen auf dem *opsi-configserver*, ob für diesen Client ein Update installiert werden soll.

Soll Software installiert werden, wird das skriptgesteuerte Installationsprogramm *opsi-winst* gestartet. Auf einer Datei-freigabe, dem sogenannten *opsi-depot*, stehen die dafür notwendigen Skripte und Softwarepakete bereit. Während dieser Zeit besteht für den Anwender keine Notwendigkeit und keine Möglichkeit in den Installationsprozess einzugreifen.

Um zu verhindern, dass sich ein Anwender vor dem Abschluss der Installation am System anmelden und so den Installations-Prozess stören kann, wird zusätzlich der sogenannte *opsi-Loginblocker* installiert, der eine Anmeldung erst nach Abschluss der Installationen zulässt.

Damit Softwarepakete mit dem Programm *opsi-winst* ohne Interaktion installiert werden können, müssen sie dafür vorbereitet werden. Siehe dazu das Kapitel *Einbindung eigener Software in die Softwareverteilung von opsi im opsi-getting-started* Handbuch.

6.1.2 Verzeichnisse des opsi-client-agent

Der *opsi-client-agent* installiert sich nach `%ProgramFiles%\opsi.org\opsi-client-agent`.

Dieses Verzeichnis enthält alle Programme des *opsi-client-agent* wie z.B. den *opsiclientd*, die *opsiclientd notifier* den *opsi-winst* und einige Bibliotheken. Weiterhin finden sich hier die Konfigurationsdateien und grafischen Vorlagen der genannten Programme.

Das Verzeichnis `%ProgramFiles%\opsi.org\opsi-client-agent` ist gegen Veränderung mit Benutzerrechten geschützt.

Das Verzeichnis `%ProgramFiles%\opsi.org\opsi-client-agent\opsiclientd` enthält die Konfigurationsdatei des *opsiclientd* und kann nur mit Administratorrechten gelesen werden.

Weiterhin gibt es das Verzeichnis `c:\opsi.org`.

Dieses Verzeichnis dient zur Zeit für den Installationscache (wenn gecached installiert wird → WAN-Erweiterung). Es wird in Zukunft noch weitere Funktionen übernehmen.

Das Verzeichnis `c:\opsi.org` kann nur mit Administratorrechten gelesen werden.

Logdateien des *opsi-client-agent* befinden sich unter `c:\opsi.org\log\`.

6.1.3 Der Service: *opsiclientd*

Der *opsiclientd* ist die Basis des *opsi-client-agents*. Er läuft als Service mit administrativen Rechten und wird beim Boot automatisch gestartet.

Wesentliche Funktionen sind:

- Eventbasierte Steuerung: Es kann auf unterschiedliche Events im System reagiert werden. Ein Event ist zum Beispiel der Start des Betriebssystems.
- Steuerung über Webservice: Auf den Webservice des *opsiclientd* kann über das Netzwerk zugegriffen werden. Diese Schnittstelle dient zum Anstoßen von Installationen (*push*) aber auch zu Wartungszwecken.
- Remote Konfiguration: Alle wesentlichen Konfigurationsdaten des *opsiclientd* lassen sich zentral über die *Hostparameter* global oder Client-spezifisch bearbeiten.

Der *opsi-client-agent* besteht aus mehreren Komponenten:

- *opsiclientd*: Der zentrale Service des *opsi-client-agents*.
- *opsiclientd notifier*: Fenster zur Information / Kommunikation mit dem Anwender
- *opsi-Loginblocker*: Sperrt den Login bis die Installationen abgeschlossen sind.

Installation

Im Rahmen einer Neuinstallation eines Betriebssystems per unattended Setup über opsi wird der *opsi-client-agent* automatisch mit installiert.

Zur Deinstallation kann der *opsi-client-agent* auf *uninstall* gesetzt werden.

Zur nachträglichen Installation oder zu Reparaturzwecken siehe Kapitel Abschnitt [6.1.6](#).

opsiclientd

Kernkomponente des *opsi-client-agents* ist der Service *opsiclientd*. Dieser läuft beim Start des Betriebssystems an.

Er übernimmt folgende Aufgaben:

- Während das System bootet und der *opsiclientd* auf den Start der Windows GUI wartet, wird der *block_login_notifier* ausgeführt. Dieser zeigt standardmäßig ein Schloss in der oberen rechten Ecke des Bildschirms.
- Er baut beim Auftreten der konfigurierten Events Kontakt zum *opsi-configserver* auf. Konfigurationen und anstehende *Action-Requests* werden per JSON-RPC abgefragt. Das Standard-Event ist hierbei *gui_startup*, welches beim Start des Rechners (Start der GUI) und damit vor dem Login aktiv wird.
- Er stellt eine Named-Pipe bereit, über die der *opsi-Loginblocker* Kontakt zu ihm aufnehmen kann, um den Zeitpunkt der Freigabe des Logins zu erfragen. Auch diese Kommunikation erfolgt per *JSON-RPC*.
- Startet den *opsiclientd notifier* zur Interaktion und Kommunikation mit dem Anwender.
- Bei Bedarf stellt er eine Verbindung zum *opsi-depot* her, aktualisiert die lokale Installation des *opsi-winst* und startet diesen zur Bearbeitung der anstehenden *Action-Requests* (Installationen).

opsiclientd notifier

Der *opsiclientd notifier* realisiert die Interaktion mit dem Anwender. Hier werden sowohl Statusmeldungen des *opsiclientd* ausgegeben als auch Dialoge, die zur Steuerung des *opsiclientd* dienen. Die jeweilige Funktion und das Erscheinungsbild wird hierbei über Konfigurations-Dateien bestimmt.

Der *opsiclientd notifier* kann zu unterschiedlichen Situationen und auf unterschiedliche Weise erscheinen:

blocklogin notifier

Wird angezeigt während der *opsi-Loginblocker* aktiv ist.



Abbildung 57: opsiclientd blocklogin notifier

event notifier

Startet beim Auftreten eines Events, gibt Informationen zum Event-Ablauf aus und bietet die Möglichkeit, die Bearbeitung eines Events abzubrechen.



Abbildung 58: opsiclientd event notifier

action notifier

Wird gestartet wenn Aktionen ausgeführt werden sollen und bietet die Möglichkeit, diese zu verschieben.

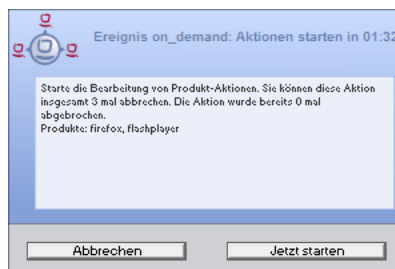


Abbildung 59: opsiclientd action notifier

shutdown notifier

Startet sobald ein Shutdown/Reboot ausgeführt werden muss und bietet die Möglichkeit, diesen zu verschieben.

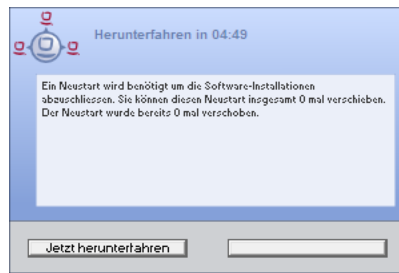


Abbildung 60: opsiclientd shutdown notifier



Achtung

Änderung der Benennung/Funktionalität von opsi 4.0.1 gegenüber opsi 4.0.

Den opsi 4.0 event notifier gibt es nicht mehr.

Der opsi 4.0.1 event notifier entspricht dem opsi 4.0 action notifier.

Der opsi 4.0.1 action notifier ähnelt dem opsi 4.0 event notifier, wird aber nur aktiv, wenn die Bearbeitung von Aktionen ansteht.

opsi-Loginblocker

Der *opsi-Loginblocker* für NT5 Win2K/WinXP ist als *GINA* implementiert (*opsigina.dll*). Diese *GINA* wartet bis zum Abschluss der *Produktaktionen* oder dem Timeout (Standard-Wert: 120 Sekunden) bei nicht erreichbarem *opsiclientd*. Danach wird die Kontrolle an die nächste *GINA* übergeben (in der Regel an die *msgina.dll*).

Der *opsi-Loginblocker* für NT6 (Vista/Win7) ist als *credential provider filter* realisiert (*OpsiLoginBlocker.dll*). Er blockiert alle *credential provider* bis zum Abschluss der *Produktaktionen* oder dem Timeout (Standard-Wert: 120 Sekunden) bei nicht erreichbarem *opsiclientd*.

Event-Ablauf

Der Ablauf der Aktionen, die in einem Event stattfinden, ist vielfältig konfigurierbar. Um die Konfigurationsmöglichkeiten zu verstehen, ist ein Verständnis der Ablauf-Logik notwendig. Es folgt zunächst ein Überblick über den Ablauf eines "Standard-Events" bei dem der opsi-configserver gefragt wird, ob Aktionen auszuführen sind (z.B. *event_gui_startup*).

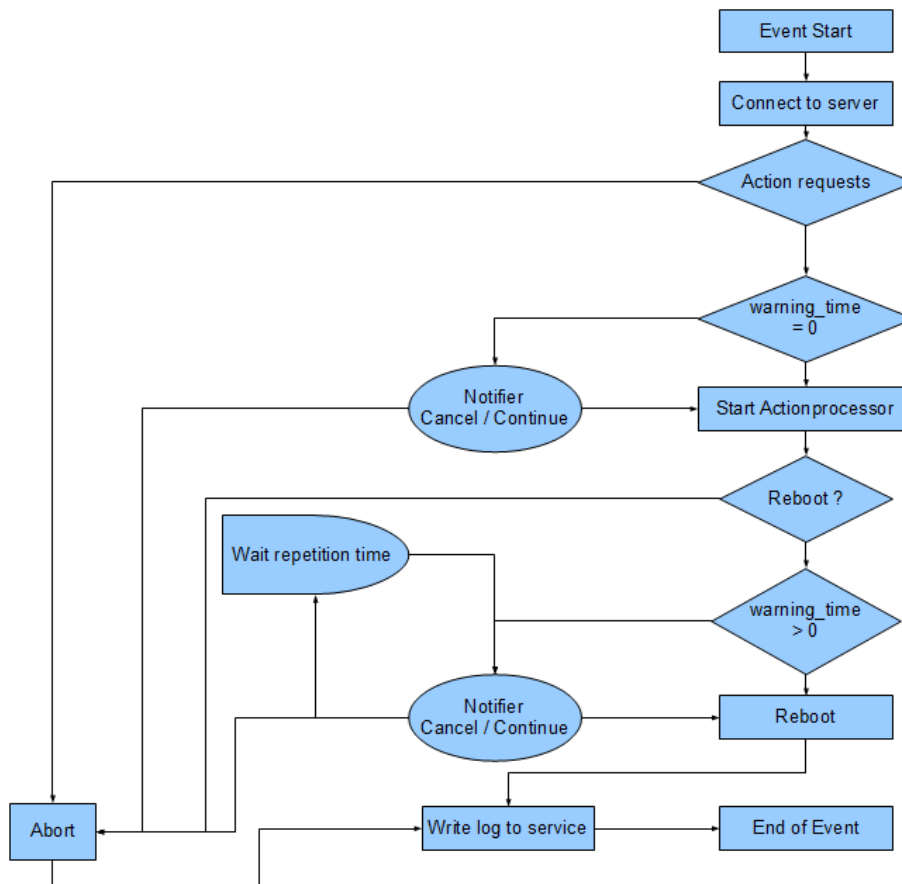


Abbildung 61: Ablauf eines Standard-Events

Die wichtigsten Parameter wirken hier wie folgt zusammen:

Tipp

Tritt bei der Verbindungsaufnahme zum *opsi-configserver* ein Fehler auf, kann natürlich auch keine Log-Datei zum *opsi-configserver* übertragen werden. Die genaue Fehlerbeschreibung ist jedoch in der *opsiclientd.log* im Log-Verzeichnis auf dem Client festgehalten.

1. Tritt ein Event ein, wird der `event_notifier_command` ausgeführt.
Nun wird versucht die konfigurierten *opsi-configserver* über deren URLs zu erreichen.
Konnte nach `user_cancelable_after` Sekunden keine Verbindung hergestellt werden, so wird im *opsiclientd* *notifier* der Button aktiviert, der das Abbrechen der Verbindungsaufnahme ermöglicht. Sobald die Verbindung zum *opsi-configserver* hergestellt ist, ist ein Abbrechen nicht mehr möglich.
Kann innerhalb von `connection_timeout` Sekunden keine Verbindung zum *opsi-configserver* hergestellt werden, so wird das laufende Event mit einem Fehler beendet. Soll der User keine Möglichkeit zum Abbrechen haben, muss `user_cancelable_after` auf einen Wert größer oder gleich `connection_timeout` gesetzt werden.
2. Wird der *opsi-configserver* erreicht, wird geprüft, ob Aktionen gesetzt sind. Sollen Aktionen ausgeführt werden wird der `action_notifier_command` ausgeführt.

Dieser *opsiclientd notifier* zeigt die Liste der Produkte an, für die Aktionen gesetzt sind und ist `action_warning_time` Sekunden sichtbar. Ist die `action_warning_time` = 0 (Standard-Wert) wird kein `action_notifier_command` ausgeführt.

Zusätzlich kann dem Anwender ermöglicht werden, das Bearbeiten der Aktionen auf einen späteren Zeitpunkt zu verschieben. Die Aktionen können hierbei `action_user_cancelable` mal verschoben werden.

Nach Erreichen der maximalen Abbrüche oder im Fall von `action_user_cancelable` = 0 kann der Anwender die Aktionen nicht verhindern.

In jedem Fall wird ein Button angezeigt, mit dem die Wartezeit abgebrochen und die Bearbeitung der Aktionen ohne weitere Verzögerung begonnen werden kann. Der Hinweis-Text, der im *opsiclientd notifier* erscheint, ist über die Option `action_message` bzw. `action_message[lang]` konfigurierbar.

Innerhalb dieses Textes können die Platzhalter `%action_user_cancelable%` (Gesamtanzahl der möglichen Abbrüche) und `%action_cancel_counter%` (Anzahl der bereits erfolgten Abbrüche) verwendet werden.

Wurden die Aktionen nicht vom User abgebrochen, wird der `action_cancel_counter` zurückgesetzt und der *opsi-winst* startet mit deren Bearbeitung.

3. Beendet sich der *opsi-winst* mit einer Reboot-/Shutdown-Anforderung so wird geprüft ob ein `shutdown_notifier_command` gesetzt ist und ob sie `shutdown_warning_time` > 0 ist. Sind diese Bedingungen erfüllt, wird der `shutdown_notifier_command` ausgeführt.

Der nun startende *opsiclientd notifier* kündigt den Reboot / Shutdown an und ist `shutdown_warning_time` Sekunden sichtbar.

Die maximale Anzahl, wie oft ein Reboot/Shutdown vom Benutzer verschoben werden kann, wird hierbei über `shutdown_user_cancelable` konfiguriert.

In jedem Fall bietet der *opsiclientd notifier* die Möglichkeit, den Shutdown/Reboot sofort auszuführen.

Bei einem Verschieben der Reboot-/Shutdown-Anforderung durch den Benutzer erscheint der *opsiclientd notifier* nach `shutdown_warning_repetition_time` Sekunden wieder.

Der Hinweis-Text ist über `shutdown_warning_message` bzw. `shutdown_warning_message[lang]` konfigurierbar.

Innerhalb dieses Textes können die Platzhalter `%shutdown_user_cancelable%` (Gesamtanzahl der möglichen Abbrüche) und `%shutdown_cancel_counter%` (Anzahl der bereits erfolgten Abbrüche) verwendet werden.

Nach erfolgtem Shutdown oder Reboot wird der `shutdown_cancel_counter` zurückgesetzt.

Tipp

Der Ablauf des Event und auch die Aktionen des Benutzers sind in der Timeline auf der Info-Seite des *opsiclientds* sichtbar (siehe Abschnitt [6.1.3](#)).

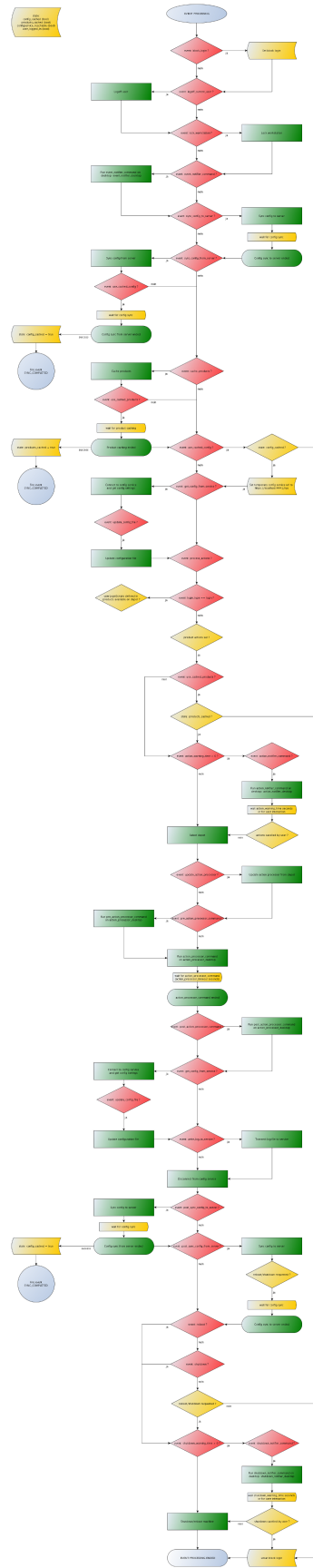


Abbildung 62: Vollständiges Ablaufdiagramm eines Events

Konfiguration

Im folgenden wird die Konfiguration des opsi-client-agent vorgestellt.

Konfiguration unterschiedlicher Events

Um den vielen unterschiedlichen Situationen gerecht zu werden, in denen der *opsi-client-agent* aktiv werden kann, sind die Konfigurations-Möglichkeiten vielfältig.

In der Konfiguration des *opsiclientd* leitet eine Sektion in der Form `[event_<config-id>]` eine neue Event-Konfiguration ein.

Eine Event-Konfiguration kann über das Setzen der Option `active = false` deaktiviert werden. Existiert zu einem Event-Typ keine Event-Konfiguration (oder sind diese deaktiviert), wird der entsprechende Event-Typ komplett deaktiviert.

Es gibt verschiedene Typen von Event-Konfigurationen (`type`).

- Es gibt *Event-Konfigurations-Vorlagen* (`type = template`)
Event-Konfigurationen können voneinander "erben". Ist über die Option `super` die Id einer anderen Event-Konfiguration gesetzt, erbt die Event-Konfiguration alle Optionen (bis auf `active`) der Parent-Konfiguration. Geerbte Optionen können jedoch überschrieben werden.
Das Deaktivieren von Events beeinflusst die Vererbung nicht.
- Alle weiteren Event-Konfigurationen gelten für einen gewissen Event-Typ (`type`).
Verfügbare Event-Typen sind:
 - `gui startup`
Ein Event vom Typ `gui startup` tritt beim Start des Clients (der GUI) auf.
Es ist das gängigste Event und ist in der Standard-Konfiguration aktiv.
 - `custom`
Event-Konfigurationen vom Typ `custom` können selbst festlegen, wann ein solches Event erzeugt wird. Hierfür kann über die Option `wql` ein *WQL*-Ausdruck angegeben werden. Sobald dieser *WQL*-Ausdruck ein Ergebnis liefert, wird ein `custom`-Event mit der jeweiligen Konfiguration gestartet.
Wird bei einem `custom`-Event die Option `wql` leer angegeben, tritt dieses Event praktisch nie auf, kann aber über die Webservice-Schnittstelle des *opsiclientd* bei Bedarf ausgelöst werden.
 - `user login`
Wird ausgelöst, wenn sich ein Benutzer am System anmeldet.
 - `timer`
Tritt in festen Intervallen auf (alle `interval` Sekunden).
 - `sync completed`
Wird ausgelöst, wenn die Synchronisation von Konfigurationen (`sync_config_from_server`) oder von Produkten (`cache_products`) erfolgt.
 - `sw on demand`
Tritt auf, wenn ein Benutzer bei Verwendung des *Software-On-Demand-Moduls* *Aktionen sofort ausführen* wählt.
- Es gibt *Preconditions* (Vorbedingungen)
Preconditions geben bestimmte Systemzustände vor (z.B. ob gerade ein Benutzer am System angemeldet ist). In der Konfiguration des *opsiclientd* leitet eine Sektion in der Form `[precondition_<precondition-id>]` die Deklaration einer *Precondition* ein. Eine *Precondition* ist dann erfüllt, wenn alle angegebenen Optionen erfüllt sind. Eine nicht angegebene Option gilt hierbei als erfüllt. Mögliche Optionen für *Preconditions* sind:
 - `user_logged_in`: ist erfüllt, wenn ein Benutzer am System angemeldet ist.
 - `config_cached`: ist erfüllt, wenn das Cachen von Konfigurationen abgeschlossen ist (siehe: `sync_config_from_server`).
 - `products_cached`: ist erfüllt, wenn das Cachen von Produkten abgeschlossen ist (siehe: `cache_products`).

- Einer Event-Konfiguration kann eine *Precondition* zugewiesen werden. Zu einer Event-Konfiguration mit *Precondition* muss immer eine entsprechende Event-Konfiguration ohne *Precondition* existieren. Hierbei erbt die Event-Konfiguration mit *Precondition* automatisch von der Event-Konfiguration ohne *Precondition*. Beim Auftreten eines Events wird nun entschieden welche *Preconditions* erfüllt sind. Ist keine der *Preconditions* erfüllt, gilt die Event-Konfiguration ohne *Precondition*. Ist eine der *Preconditions* erfüllt, gilt die Event-Konfiguration die mit dieser *Precondition* verknüpft ist. Sind mehrere *Preconditions* erfüllt, so wird die *Precondition* bevorzugt, die am genauesten definiert ist (die meisten Optionen besitzt).

Ein Beispiel zur Erläuterung:

Im Rahmen einer Installation kann es notwendig sein den Rechner zu rebooten. Ist gerade ein Benutzer am System angemeldet, sollte dieser über den anstehenden Reboot informiert werden. Hierbei ist eine angemessene Wartezeit vor dem Ausführen des Reboots angebracht. Zusätzlich kann es sinnvoll sein, dem Benutzer die Entscheidung zu überlassen, ob der Reboot besser zu einem späteren Zeitpunkt ausgeführt werden soll.

Ist zum Zeitpunkt des benötigten Reboots jedoch kein Benutzer angemeldet, ist es sinnvoll, den Reboot ohne weitere Wartezeit sofort durchzuführen.

Dieses Problem wird am Beispiel von `event_on_demand` wie folgt konfiguriert:

- Es wird eine *Precondition* `user_logged_in` definiert, die erfüllt ist, wenn ein Benutzer am System angemeldet ist (`user_logged_in = true`).
- In der Event-Konfiguration `event_on_demand` (ohne *Precondition*) wird `shutdown_warning_time = 0` gesetzt (sofortiger Reboot ohne Meldung).
- In der Event-Konfiguration `event_on_demand{user_logged_in}` wird `shutdown_warning_time = 300` gesetzt (300 Sekunden Vorwarnzeit).

Proxysupport-Konfiguration

In der global-Sektion von der `opsiclientd.conf` gibt es jetzt die Möglichkeit, den `opsi-client-agent` einen Proxyserver mit zu konfigurieren. Wenn ein Proxy konfiguriert wurde, werden alle HTTP- und HTTPS-Verbindungen vom `opsiclientd` über diesen Proxy umgeleitet.

```
# Use a proxy for connecting configservice
# proxy_mode:
# 'system' will try to check the system setting,
# 'static' to use proxyurl from configfile/hostparameter
# proxy_url usage: http://<user>:<password>@<proxy-url>:<proxy-port>
# Example: http://proxyuser:proxypass123@proxy.domain.local:8080
proxy_mode = static
proxy_url =
```

Die Proxyeinstellungen erlauben auch einen Proxy zu benutzen, der eine Authentifizierung erfordert. Dazu muss die `Proxy_url` wie oben im Beispiel angegeben werden.



Warnung

Der `proxy_mode` ist vorgesehen, dass bei der Einstellung `system`, der proxy aus dem laufenden Client-System ausgelesen werden. Dies ist im Moment nicht implementiert, deshalb funktioniert momentan nur die Einstellung `static`.

Steuerung der Produkte die ausgeführt werden pro Event

Mit diesem neuen Feature ist es über die Konfiguration möglich, die Liste der ab zu bearbeitenden Produkte über Produktgruppen zu steuern.

Dazu gibt es Grundsätzlich zwei Vorgehensweise:

Blacklisting (Ausschliessen):

Mit der Option `exclude_product_group_ids` kann man nun eine Kommaseparierte Liste von Produktgruppen-Ids mitgeben, dessen Mitglieder vom aktuellen Event ausgeschlossen werden. Auch wenn Sie eigentlich auf setup stehen. Diese Produkte werden zwar ignoriert, aber bleiben auf setup stehen.

Whitelisting (Liste von Produkten ausschliesslich freigeben):

Mit der Option `include_product_group_ids` kann man nun eine Kommaseparierte Liste von Produktgruppen-Ids festlegen, dessen Mitglieder überhaupt bearbeitet werden dürfen, vorausgesetzt Sie eine Aktion ist auch gesetzt.

Diese Einstellung kann man entweder Global im Default-Event angeben, damit das für jedes Event gilt. Man kann diese Optionen aber auch Zum Beispiel nur im Event `on_demand` einsetzen, somit kann man Pakete die auf setup stehen von Push-Installationen ausschliessen, obwohl Sie auf setup stehen. Bei einem normalen Neustarts des Clients mit `gui_startup` (default) würden diese ausgeschlossenen Pakete trotzdem auf dem Client installiert werden.



Achtung

Für Clients, die das Modul WAN/VPN aktiviert haben, muss man diese Optionen neben dem Sync-Event auch in der CacheService-Sektion mit aufgenommen werden, da der CacheService zwar vom Sync-Event getriggert wird, aber selbst keinen Zugriff auf das sync-Event hat.



Warnung

Produktabhängigkeiten werden bei diesem Feature nicht berücksichtigt. Bitte achten Sie darauf, dass Sie bei der Konfiguration keine Abhängigkeiten ausser Kraft setzen.

Konfiguration über die Konfigurationsdatei

Die Konfigurationsdatei ist auf einem 64Bit Windows zu finden unter `c:\program files (x86)\opsi.org\opsi-client-agent\opsiclientd\opsiclientd.conf`.

Auf einem 32Bit Windows ist die Konfigurationsdatei unter `c:\program files\opsi.org\opsi-client-agent\opsiclientd` zu finden.



Achtung

Diese Konfigurationsdatei ist UTF-8 kodiert.

Änderungen mit Editoren, die diese Kodierung nicht beherrschen (z.B. notepad.exe), zerstören die Umlaute in dieser Datei.

Die hier festgelegte Konfiguration kann nach erfolgreicher Verbindung zum *opsi-configserver* durch die dort festgelegte *Hostparameter* überschrieben werden. Beispiel `opsiclientd.conf`:

```
; = = = = =
; =   configuration file for opsiclientd   =
; = = = = =

; - - - - -
; -   global settings   -
; - - - - -

[global]

# Location of the log file.
log_file = c:\\opsi.org\\log\\opsiclientd.log

# Set the log (verbosity) level
# (0 <= log level <= 9)
```

```

# 0: nothing, 1: essential, 2: critical, 3: errors, 4: warnings, 5: notices
# 6: infos, 7: debug messages, 8: more debug messages, 9: passwords
log_level = 4

# Client id.
host_id =

# Opsi host key.
opsi_host_key =

# Verify opsi server certs
verify_server_cert = false

# Verify opsi server certs by ca
verify_server_cert_by_ca = false

# On every daemon startup the user login gets blocked
# If the gui starts up and no events are being processed the login gets unblocked
# If no gui startup is noticed after <wait_for_gui_timeout> the login gets unblocked
# Set to 0 to wait forever
wait_for_gui_timeout = 120

# Application to run while blocking login
block_login_notifier = %global.base_dir%\notifier.exe -s notifier\block_login.ini

# Use a proxy for connecting configservice
# proxy_mode:
#   'system' will try to check the system setting,
#   'static' to use proxyurl from configfile/hostparameter
# proxy_url usage: http://<user>:<password>@<proxy-url>:<proxy-port>
# Example: http://proxyuser:proxypass123@proxy.domain.local:8080
proxy_mode = static
proxy_url =

; - - - - -
; -   config service settings   -
; - - - - -
[config_service]
# Service url.
# http(s)://<opsi config server address>:<port>/rpc
url = https://opsi.uib.local:4447/rpc

# Connection timeout.
connection_timeout = 30

# The time in seconds after which the user can cancel the connection establishment
user_cancelable_after = 30

# If this option is set, the local system time will be synced with time from service
sync_time_from_service = false

; - - - - -
; -   depot server settings   -
; - - - - -
[depot_server]

# Depot server id
depot_id =

# Depot url.
# smb://<depot address>/<share name>/<path to products>
url =

# Local depot drive
drive =

# Username that is used for network connection [domain\]<username>
username = pcpatch

```

```
; -----
; -   cache service settings   -
; -----
[cache_service]
# Maximum product cache size in bytes
product_cache_max_size = 5000000000
# Members of this ProductGroups will be excluded from processing
exclude_product_group_ids =
# Only members of this ProductGroups will be excluded from processing
include_product_group_ids =

; -----
; -   control server settings   -
; -----
[control_server]

# The network interfaces to bind to.
# This must be the IP address of a network interface.
# Use 0.0.0.0 to listen to all interfaces
interface = 0.0.0.0

# The port where opsiclientd will listen for HTTPS rpc requests.
port = 4441

# The location of the server certificate.
ssl_server_cert_file = %global.base_dir%\opsiclientd\opsiclientd.pem

# The location of the server private key
ssl_server_key_file = %global.base_dir%\opsiclientd\opsiclientd.pem

# The location of the static files
static_dir = %global.base_dir%\opsiclientd\static_html

# The maximum number of authentication failures before a client ip
# is blocked for an amount of time.
max_authentication_failures = 5

; -----
; -   notification server settings   -
; -----
[notification_server]

# The network interfaces to bind to.
# This must be the IP address of a network interface.
# Use 0.0.0.0 to listen to all interfaces
interface = 127.0.0.1

# The first port where opsiclientd will listen for notification clients.
start_port = 44000

# Port for popup notification server
popup_port = 45000

; -----
; -   opsiclientd notifier settings   -
; -----
[opsiclientd_notifier]

# Notifier application command
command = %global.base_dir%\notifier.exe -p %port% -i %id%

; -----
; -   opsiclientd rpc tool settings   -
; -----
[opsiclientd_rpc]

# RPC tool command
```



```

command = %global.base_dir%\opsiclientd_rpc.exe "%global.host_id%" "%global.opsi_host_key%" "%control_server.port%"

; -----
; -   action processor settings                               -
; -----
[action_processor]
# Locations of action processor
local_dir = %global.base_dir%\opsi-winst
remote_dir = opsi-winst\files\opsi-winst
filename = winst32.exe

# Action processor command
command = "%action_processor.local_dir%\%action_processor.filename%" /opiservice "%service_url%" /clientid %global.\
    host_id% /username %global.host_id% /password %global.opsi_host_key%

# Load profile / environment of %run_as_user%
create_environment = false

; -----
; -   events                                                 -
; -----
[event_default]
; === Event configuration
# Type of the event (string)
type = template
# Interval for timer events in seconds (int)
interval = -1
# Maximum number of event repetitions after which the event will be deactivated (int, -1 = forever)
max_repetitions = -1
# Time in seconds to wait before event becomes active (int, 0 to disable delay)
activation_delay = 0
# Time in seconds to wait before an event will be fired (int, 0 to disable delay)
notification_delay = 0
# Event notifier command (string)
event_notifier_command = %opsiclientd_notifier.command% -s notifier\event.ini
# The desktop on which the event notifier will be shown on (current/default/winlogon)
event_notifier_desktop = current
# Block login while event is been executed (bool)
block_login = false
# Lock workstation on event occurrence (bool)
lock_workstation = false
# Logoff the current logged in user on event occurrence (bool)
logoff_current_user = false
# Get config settings from service (bool)
get_config_from_service = true
# Store config settings in config file (bool)
update_config_file = true
# Transmit log file to opsi service after the event processing has finished (bool)
write_log_to_service = true
# Shutdown machine after action processing has finished (bool)
shutdown = false
# Reboot machine after action processing has finished (bool)
reboot = false
# Members of this ProductGroups will be excluded from processing
exclude_product_group_ids =
# Only members of this ProductGroups will be excluded from processing
include_product_group_ids =

; === Sync/cache settings
# Sync configuration from local config cache to server (bool)
sync_config_to_server = false
# Sync configuration from server to local config cache (bool)
sync_config_from_server = false
# Sync configuration from local config cache to server after action processing (bool)
post_sync_config_to_server = false
# Sync configuration from server to local config cache after action processing (bool)
post_sync_config_from_server = false
# Work on local config cache

```

```

use_cached_config = false
# Cache products for which actions should be executed in local depot cache (bool)
cache_products = false
# Maximum transfer rate when caching products in byte/s (int, 0 = no limit)
cache_max_bandwidth = 0
# Dynamically adapt bandwidth to other network traffic (bool)
cache_dynamic_bandwidth = false
# Work on local depot cache
use_cached_products = false

; === Action notification (if product actions should be processed)
# Time in seconds for how long the action notification is shown (int, 0 to disable)
action_warning_time = 0
# Action notifier command (string)
action_notifier_command = %opsiclientd_notifier.command% -s notifier\\action.ini
# The desktop on which the action notifier will be shown on (current/default/winlogon)
action_notifier_desktop = current
# Message shown in the action notifier window (string)
action_message = Starting to process product actions. You are allowed to cancel this event a total of \%
    action_user_cancelable% time(s). The event was already canceled %state.action_processing_cancel_counter% time(s).
# German translation (string)
action_message[de] = Starte die Bearbeitung von Produkt-Aktionen. Sie können diese Aktion insgesamt \%
    action_user_cancelable% mal abbrechen. Die Aktion wurde bereits %state.action_processing_cancel_counter% mal \
    abgebrochen.
# French translation (string)
action_message[fr] = Traitement des actions du produit. Vous êtes autorisé à annuler cet événement un total de \%
    action_user_cancelable% fois. L'événement a été déjà annulée %state.action_processing_cancel_counter% fois.
# Number of times the user is allowed to cancel the execution of actions (int)
action_user_cancelable = 0

; === Action processing
# Should action be processed by action processor (bool)
process_actions = true
# Type of action processing (default/login)
action_type = default
# Update the action processor from server before starting it (bool)
update_action_processor = true
# Command which should be executed before start of action processor
pre_action_processor_command =
# Action processor command (string)
action_processor_command = %action_processor.command%
# The desktop on which the action processor command will be started on (current/default/winlogon)
action_processor_desktop = current
# Action processor timeout in seconds (int)
action_processor_timeout = 10800
# Command which should be executed before after action processor has ended
post_action_processor_command =

; === Shutdown notification (if machine should be shut down or rebooted)
# Process shutdown requests from action processor
process_shutdown_requests = true
# Time in seconds for how long the shutdown notification is shown (int, 0 to disable)
shutdown_warning_time = 0
# Shutdown notifier command (string)
shutdown_notifier_command = %opsiclientd_notifier.command% -s notifier\\shutdown.ini
# The desktop on which the action notifier will be shown on (current/default/winlogon)
shutdown_notifier_desktop = current
# Message shown in the shutdown notifier window (string)
shutdown_warning_message = A reboot is required to complete software installation tasks. You are allowed to delay this \
    reboot a total of %shutdown_user_cancelable% time(s). The reboot was already delayed %state.\
    shutdown_cancel_counter% time(s).
# German translation (string)
shutdown_warning_message[de] = Ein Neustart wird benötigt um die Software-Installationen abzuschliessen. Sie können \
    diesen Neustart insgesamt %shutdown_user_cancelable% mal verschieben. Der Neustart wurde bereits %state.\
    shutdown_cancel_counter% mal verschoben.
# French translation (string)
shutdown_warning_message[fr] = Un redémarrage est nécessaire pour terminer l'installation du logiciel. Vous êtes \
    autorisé à retarder le redémarrage un total de %shutdown_user_cancelable% fois. Le redémarrage a été déjà retardé \

```

```
%state.shutdown_cancel_counter% fois.
# Number of times the user is allowed to cancel the shutdown (int)
shutdown_user_cancelable = 0
# Time in seconds after the shutdown notification will be shown again after the user has canceled the shutdown (int)
shutdown_warning_repetition_time = 3600

[event_gui_startup]
super = default
type = gui startup
name = gui_startup
block_login = true

[event_gui_startup{user_logged_in}]
name = gui_startup
shutdown_warning_time = 300
block_login = false

[event_gui_startup{cache_ready}]
use_cached_config = true
use_cached_products = true
action_user_cancelable = 3
action_warning_time = 60

[event_gui_startup{installation_pending}]
name = gui_startup
active = true

[event_on_demand]
super = default
type = custom
name = on_demand

[event_on_demand{user_logged_in}]
name = on_demand
shutdown_warning_time = 300

[event_software_on_demand]
super = default
type = sw on demand

[event_sync]
super = default
type = template
process_actions = false
event_notifier_command =
sync_config_to_server = true
sync_config_from_server = true
cache_products = true
cache_dynamic_bandwidth = true

[event_timer]
super = sync
type = timer
active = false
interval = 3600

[event_net_connection]
super = sync
type = custom
active = false
wql = SELECT * FROM __InstanceModificationEvent WITHIN 2 WHERE TargetInstance ISA 'Win32_NetworkAdapter' AND \
      TargetInstance.NetConnectionStatus = 2

[event_sync_completed]
super = default
type = sync completed
event_notifier_command =
process_actions = false
```

```
get_config_from_service = false
write_log_to_service = false

[event_sync_completed{cache_ready_user_logged_in}]
reboot = true
shutdown_user_cancelable = 10
shutdown_warning_time = 300

[event_sync_completed{cache_ready}]
reboot = true

[event_user_login]
super = default
type = user login
action_type = login
active = false
action_message = Starting to process user login actions.
action_message[de] = Beginne mit der Verarbeitung der Benutzer-Anmeldungs-Aktionen.
action_message[fr] = Traitement des actions à la connexion de l'utilisateur.
block_login = false
process_shutdown_requests = false
get_config_from_service = false
update_config_file = false
write_log_to_service = false
update_action_processor = true
event_notifier_command = %opsiclientd_notifier.command% -s notifier\\userlogin.ini
event_notifier_desktop = default
action_processor_command = %action_processor.command% /sessionid %service_session% /alloginscripts /silent
action_processor_desktop = default
action_processor_timeout = 300

[event_on_shutdown]
super = default
type = custom
name = on_shutdown
active = False

[event_on_shutdown{installation_pending}]
name = on_shutdown
active = False

[event_silent_install]
super = default
type = custom
name = silent_install
event_notifier_command =
process_shutdown_requests = false
action_processor_productIds = swaudit,hwaudit
action_processor_command = %action_processor.command% /productlist %action_processor_productIds% /silent
action_processor_desktop = winlogon
action_processor_timeout = 300

[event_timer_silentinstall]
super = silent_install
type = timer
active = false
interval = 21600

[precondition_user_logged_in]
user_logged_in = true

[precondition_cache_ready]
config_cached = true
products_cached = true

[precondition_cache_ready_user_logged_in]
user_logged_in = true
config_cached = true
```

```
products_cached = true

[precondition_installation_pending]
installation_pending = true
```

Konfiguration über den Webservice (Hostparameter)

Die Konfiguration kann auch zentral gesteuert werden. Hierzu dienen Einträge in der *Hostparameter* des *opsi-configservers*.

Diese Einträge müssen dem folgenden Muster folgen:
`opsiclientd.<name der section>.<name der option>`

Ein Beispiel:

```
opsiclientd.event_gui_startup.action_warning_time = 20
```

setzt in der Konfigurationsdatei `opsiclientd.conf` in der Sektion `[event_gui_startup]` den Wert von `action_warning_time` auf 20.

Die folgende Abbildung zeigt, wie diese Werte als Defaults für alle Clients über den *opsi-configd* gesetzt werden können.

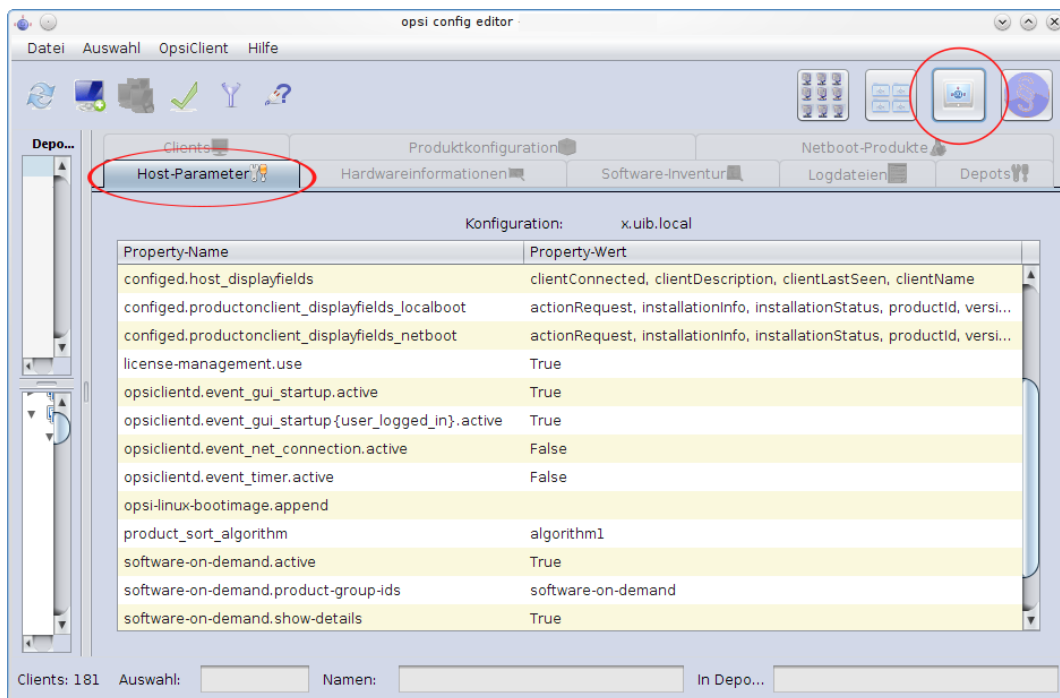


Abbildung 63: Serverweite Konfiguration des opsiclientd über den opsi-configd

Hier kann über das Kontextmenü **Property hinzufügen** ein neuer Wert gesetzt werden.

Um einen Hostparameter zu löschen, verwenden Sie das Werkzeug *opsi-admin*. Beispiel:

```
opsi-admin -d method config_delete "opsiclientd.event_gui_startup.action_warning_time"
```

Eine Client-spezifische Änderung über den *opsi-configd* führen Sie über den *Hosts-Parameter* Tab in der Client-Konfiguration aus. Um Client-spezifische Einträge zu löschen, verwenden Sie das Werkzeug *opsi-admin*. Beispiel:

```
@opsi-admin> method configState_delete "opsiclientd.event_gui_startup.action_warning_time" "myclient.uib.local"
```

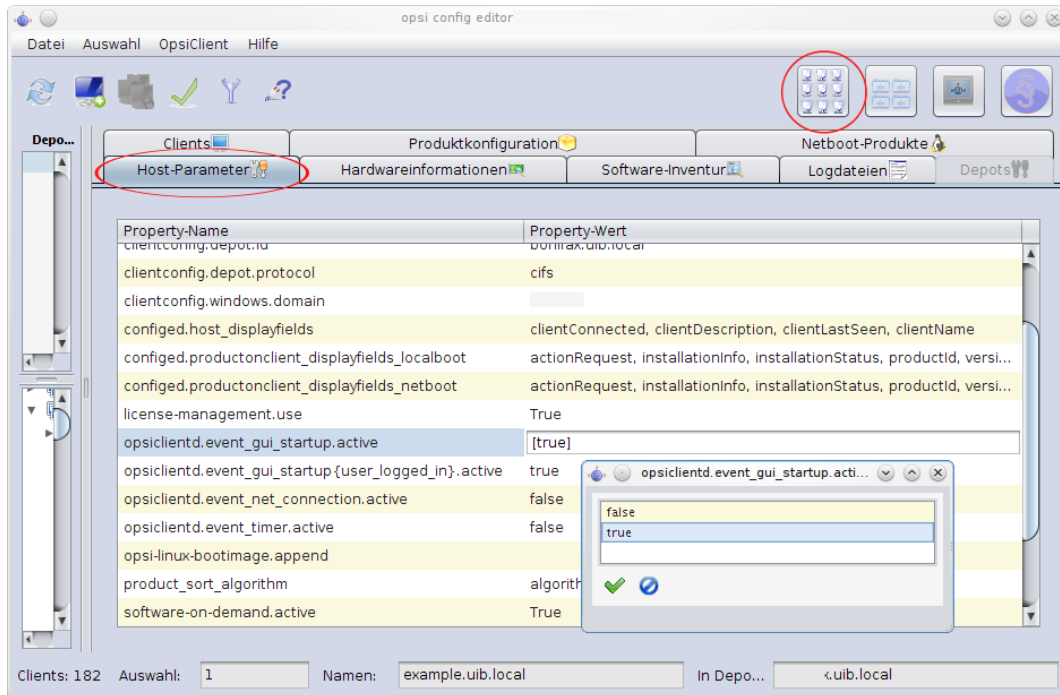


Abbildung 64: Client-spezifische Konfiguration des opsiclientd über den opsi-configed

Logging

Die Log-Datei des *opsiclientd* ist standardmäßig `c:\opsi.org\log\opsiclientd.log`.

Die Log-Informationen werden auch an den *opsi-configserver* übertragen. Dort liegen sie unter `/var/log/opsi/clientconnect/<ip-bzw.-name-des-clients>.log`. Sie sind auch im *opsi-configed* über Logdateien ⇒ Clientconnect einsehbar.

Jede Zeile in der Logdatei folgt dem Muster:

`[<log level>] [<datum zeit>] [Quelle der Meldung] Meldung (Quellcode-Datei|Zeilennummer).`

Dabei gibt es die folgenden Log-Level:

```
# Set the log (verbosity) level
# (0 <= log level <= 9)
# 0: nothing, 1: essential, 2: critical, 3: errors, 4: warnings, 5: notices
# 6: infos, 7: debug messages, 8: more debug messages, 9: passwords
```

Beispiel:

```
(...)
[5] [Mar 22 10:17:46] [ event processing gui_startup ] Event config 'sync_completed{cache_ready}' added to event \
generator 'sync_completed' (Events.pyo|1107)
[5] [Mar 22 10:17:46] [ event processing gui_startup ] Event config 'gui_startup' added to event generator '\
gui_startup' (Events.pyo|1107)
[5] [Mar 22 10:17:46] [ event processing gui_startup ] Event config 'gui_startup{cache_ready}' added to event \
generator 'gui_startup' (Events.pyo|1107)
[5] [Mar 22 10:17:46] [ event processing gui_startup ] Event config 'on_demand' added to event generator 'on_demand' \
(Events.pyo|1107)
[5] [Mar 22 10:17:46] [ event processing gui_startup ] Event config 'sync_completed{cache_ready_user_logged_in}' added\
to event generator 'sync_completed' (Events.pyo|1107)
[5] [Mar 22 10:17:46] [ event processing gui_startup ] Event config 'gui_startup{user_logged_in}' added to event \
generator 'gui_startup' (Events.pyo|1107)
[5] [Mar 22 10:17:46] [ event processing gui_startup ] Event config 'sync_completed' added to event generator '\
sync_completed' (Events.pyo|1107)
```

```

[5] [Mar 22 10:17:46] [ event processing gui_startup ] Event config 'software_on_demand' added to event generator '\
software_on_demand' (Events.pyo|1107)
[5] [Mar 22 10:17:46] [ event processing gui_startup ] Event config 'on_demand{user_logged_in}' added to event \
generator 'on_demand' (Events.pyo|1107)
[5] [Mar 22 10:17:46] [ event processing gui_startup ] Updating config file: 'C:\Program Files (x86)\opsi.org\opsi-\
client-agent\opsiclientd\opsiclientd.conf' (Config.pyo|287)
[5] [Mar 22 10:17:46] [ event processing gui_startup ] No need to write config file 'C:\Program Files (x86)\opsi.org\
opsi-client-agent\opsiclientd\opsiclientd.conf', config file is up to date (Config.pyo|318)
[5] [Mar 22 10:17:46] [ event processing gui_startup ] No product action requests set (EventProcessing.pyo|591)
[5] [Mar 22 10:17:49] [ event processing gui_startup ] Writing log to service (EventProcessing.pyo|247)
[6] [Mar 22 10:17:49] [ opsiclientd ] shutdownRequested: 0 (Windows.pyo|340)
[6] [Mar 22 10:17:49] [ opsiclientd ] rebootRequested: 0 (Windows.pyo|326)
[5] [Mar 22 10:17:49] [ opsiclientd ] Block login now set to 'False' (Opsiclientd.pyo|111)
[6] [Mar 22 10:17:49] [ opsiclientd ] Terminating block login notifier app (pid 1620) (Opsiclientd.\
pyo|148)
[6] [Mar 22 10:17:49] [ event processing gui_startup ] Stopping notification server (EventProcessing.pyo|225)
[6] [Mar 22 10:17:51] [ control server ] client connection lost (Message.pyo|464)
[6] [Mar 22 10:17:52] [ event processing gui_startup ] Notification server stopped (Message.pyo|651)
[5] [Mar 22 10:17:52] [ event processing gui_startup ] ===== EventProcessingThread for event 'gui_startup' \
ended ===== (EventProcessing.pyo|1172)
[5] [Mar 22 10:17:52] [ opsiclientd ] Done processing event '<ocdlib.Events.GUIStartupEvent object at\
0x023CE330>' (Opsiclientd.pyo|405)
[5] [Mar 22 10:19:41] [ opsiclientd ] Session 'HSzMB1wt0iBS6vH17mh3ro5r6s3TanFu' from ip '127.0.0.1',\
application 'opsi jsonrpc module version 4.0.1' expired after 120 seconds (Session.pyo|184)
[6] [Mar 22 10:19:41] [ opsiclientd ] Session timer <_Timer(Thread-20, started daemon 2636)> canceled\
(Session.pyo|120)
[5] [Mar 22 10:19:41] [ opsiclientd ] Session 'HSzMB1wt0iBS6vH17mh3ro5r6s3TanFu' from ip '127.0.0.1',\
application 'opsi jsonrpc module version 4.0.1' deleted (Session.pyo|207)
[6] [Mar 22 10:27:55] [ control pipe ] Creating pipe \\.\pipe\opsiclientd (ControlPipe.pyo|253)
[5] [Mar 22 10:27:55] [ event generator wait_for_gui ] ----> Executing: getBlockLogin() (JsonRpc.pyo|123)
[5] [Mar 22 10:27:55] [ opsiclientd ] rpc getBlockLogin: blockLogin is 'False' (ControlPipe.pyo\
|428)
[6] [Mar 22 10:27:55] [ event generator wait_for_gui ] Got result (JsonRpc.pyo|131)
,

```

Die Log-Datei des *opsi-Loginblockers* befindet sich unter NT6 (Vista/Win7) als auch unter NT5 (Win2k/WinXP) in C:\opsi.org\log\opsi_loginblocker.log.

opsiclientd infopage

Da bei den Abläufen im *opsiclientd* vielfältige Komponenten zusammenwirken, welche zum Teil gleichzeitig aktiv sind, wird die Logdatei leicht unübersichtlich.

Daher verfügt der *opsiclientd* über eine eigene *infopage* welche die Abläufe auf einer Zeitachse grafisch darstellt. Diese *infopage* kann mit dem Browser über die URL <https://<adresse-des-clients>:4441/info.html> aufgerufen werden.

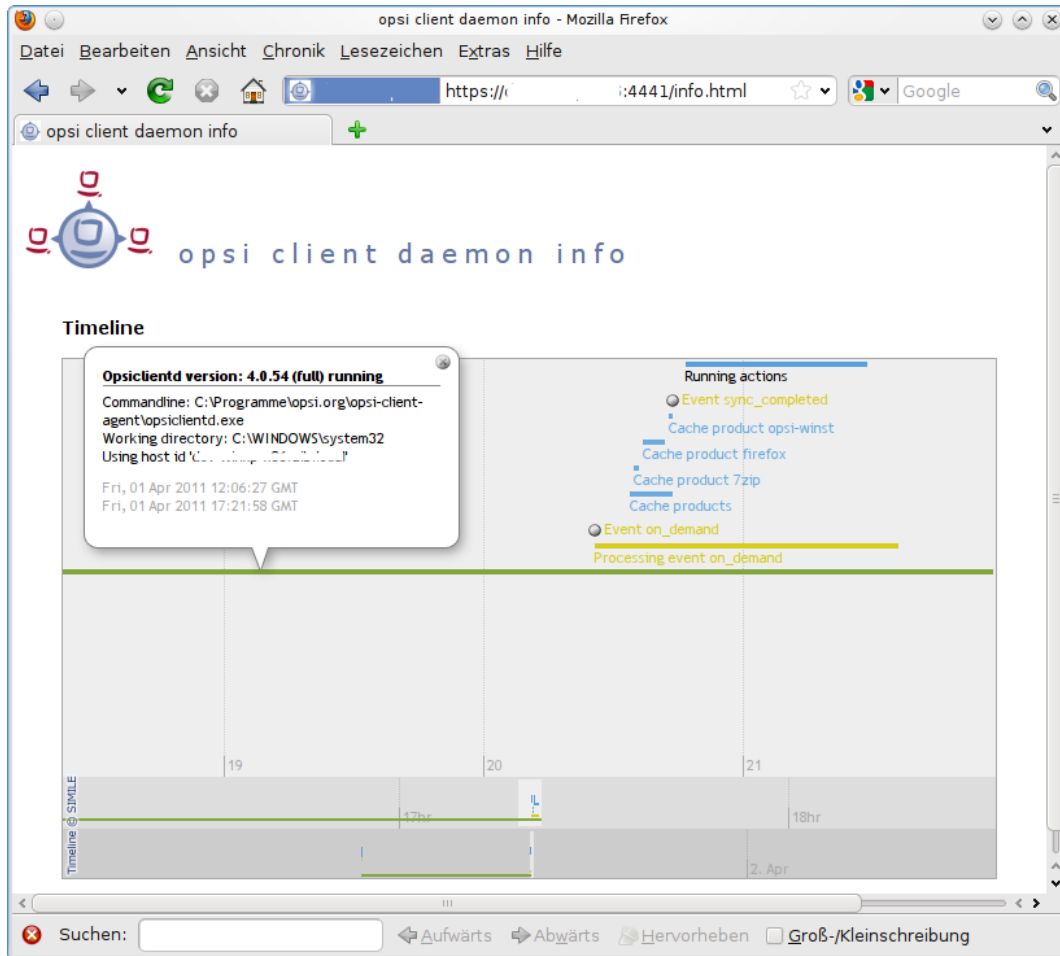


Abbildung 65: Info-Page des opsiclientd nach einer Push-Installation mit aktiviertem Produkt-Caching

Fernsteuerung des opsi-client-agent

Der *opsiclientd* verfügt über eine Webservice-Schnittstelle. Diese ermöglicht es, dem opsi-client-agent Anweisungen zu übermitteln und Vieles mehr. Sie lassen sich momentan grob in drei Bereiche aufteilen:

- Nachrichten (Popup) versenden
- *Push*-Installationen durch auslösen von Events (z.B. *on_demand*)
- Sonstige Wartungsarbeiten

Dies kann auch auf der Kommandozeile mittels Aufrufs einer *hostControlSafe_**-Methode über *opsi-admin* geschehen. Bei Verwendung der *hostControlSafe_**-Methoden `opsi-admin -d method hostControlSafe_xx *hostIds` kann der Parameter **hostIds*

- sein ["*"], dann gilt der Aufruf für alle Clients
- einen Client enthalten (z.B. "myclient.uib.local")
- eine Liste von Clients enthalten ["<client1>", "<client2>", ...]
z.B. ["client1.uib.local", "client2.uib.local"]
- eine Wildcard enthalten, wobei * als Platzhalter dient
z.B. "client.*" oder "*.uib.*"

Werden Rechner nicht erreicht (z.B. weil sie aus sind), wird für diese Rechner eine Fehlermeldung ausgegeben.

Nachrichten per Popup senden

Über den *opsi-configed* lassen sich Nachrichten an einen oder mehrere Clients versenden.

Siehe dazu Kapitel Abschnitt 4.8.6

Auf der Kommandozeile lässt sich dies ebenfalls mittels *opsi-admin* durchführen:

```
opsi-admin -d method hostControlSafe_showPopup message *hostid
```

Beispiel:

```
opsi-admin -d method hostControlSafe_showPopup "Ein Text..." "myclient.uib.local"
```

Push-Installationen: Event *on demand* auslösen

Vom *opsi-server* aus kann der Client aufgefordert werden, die gesetzten *Produktaktionen* auszuführen.

Das Auslösen des Events kann vom *opsi-configed* aus erfolgen. Abschnitt 4.8.5

Auf der Kommandozeile lässt sich dies ebenfalls mittels *opsi-admin* durchführen:

```
opsi-admin -d method hostControlSafe_fireEvent event *hostIds
```

Beispiel:

```
opsi-admin -d method hostControlSafe_fireEvent "on_demand" "myclient.uib.local"
```

Sonstige Wartungsarbeiten (shutdown, reboot, ...)

Über den Webservice des *opsiclientd* ist es möglich, steuernd auf den *opsi-client-agent* einzuwirken. Dazu muss man sich an diesem Webservice authentifizieren. Dies geschieht entweder mittels des lokalen Administrator-Accounts (ein leeres Passwort ist unzulässig) oder mittels der *opsi-host-Id* (FQDN / vollständiger Host-Name inkl. DNS-Domain) als Benutzername und des *opsi-host-Schlüssels* als Passwort.

Vom *opsi-configed* aus geht dies über das Menü *OpsIClient* oder aus dem Kontextmenü des *Client*-Tabs.

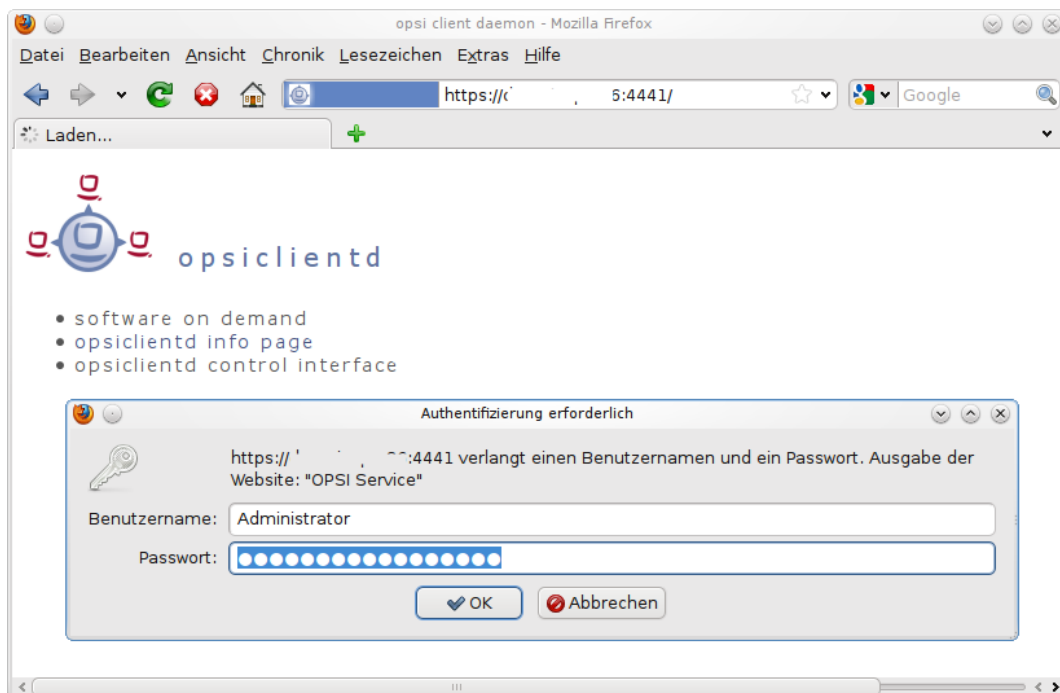


Abbildung 66: Webservice des opsciclientd

Auch auf der Kommandozeile gibt es hierfür Entsprechungen:

shutdown:

```
opsi-admin -d method hostControlSafe_shutdown [hostIds]
```

reboot:

```
opsi-admin -d method hostControlSafe_reboot [hostIds]
```

6.1.4 Anpassen des opsi-client-agent an Corporate Identity (CI)

Die Anpassung des Erscheinungsbildes des *opsi-client-agent* kann insbesondere bei der Einführung erheblich zur Akzeptanz beitragen. So kann z.B. durch das Einfügen eines bekannten Firmenlogos in die Hintergrundgrafiken eine Verunsicherung der Anwender vermieden werden.

Seit opsi-client-agent 4.0.7.16 basieren alle graphischen Komponenten des opsi-client-agent (notifier, opsi-script, kiosk-client) auf den Darstellungskomponenten zum Anzeigen von Grafiken und werden auf die selbe Weise angepasst. Farben können auf drei unterschiedliche Weise angegeben werden: Als symbolischer Name (c1Red), als Hexadezimalwert (\$FF00FF) oder als rgb Wertliste ((255,0,0)). Ein Hilfsprogramm zur Auswahl von Farben und deren richtigen Schreibweise ist der [opsi color chooser](#).

Als Hintergrund Grafikformate kommt eine Vielzahl unterschiedlicher Bitmap Formate wie .bmp, .png, jpeg usw in Frage. All dies Formate sind wieder Containerformate, dh. z.B. PNG ist nicht unbedingt gleich PNG. Evtl ist das eine Darstellbar und das andere nicht. Ein Hilfsprogramm mit dem Sie schnell prüfen können ob eine gegeben Bitmap Grafik korrekt angezeigt werden wird, ist der [opsi bitmap viewer](#).

Anzupassende Elemente: opsi-winst

Die Dateien die Sie beim opsi-winst anpassen können finden Sie im Verzeichnis `/var/lib/opsi/depot/opsi-client-agent/files/opsi/opsi-winst/winstskin`:

- `bg.png`
Die Default Hintergrundgrafik des *opsi-winst* in welche dann zur Laufzeit Textmeldungen und Produktlogos eingeblendet werden. Der Name kann in der Datei `skin.ini` angepasst werden.
- `skin.ini`
Die Konfigurationsdatei in der festgelegt ist, an welcher Stelle, mit welchem Font und Farbe Textmeldungen eingeblendet werden.

Anzupassende Elemente: opsiclientd

Im Verzeichnis `/var/lib/opsi/depot/opsi-client-agent/files/opsi/dist/notifier` finden sich die Dateien welche das Erscheinungsbild der unterschiedlichen Notifier bestimmen. Dabei gibt es für jeden Notifier eine Bild- und eine Konfigurationsdatei:

- `block_login.bmp`
Hintergrundbild des notifiers der einen aktiven Loginblocker anzeigt.
- `block_login.ini`
Konfigurationsdatei des Loginblocker notifiers.
- `event.bmp`
Hintergrundbild des notifiers der einen aktives Event mit Connection zum opsi-server anzeigt.
- `event.ini`
Konfigurationsdatei des Event notifiers.

- **action.bmp**
Hintergrundbild des notifiers der eine anstehende Aktion (Softwareinstallation) anzeigt.
- **action.ini**
Konfigurationsdatei des Action notifiers.
- **shutdown.bmp**
Hintergrundbild des notifiers der einen anstehenden Shutdown oder Reboot anzeigt.
- **shutdown.ini**
Konfigurationsdatei des Shutdown notifiers.
- **popup.bmp**
Hintergrundbild des notifiers der eine vom Server gesendete Popup Nachricht anzeigt.
- **popup.ini**
Konfigurationsdatei des Popup notifiers.
- **userlogin.bmp**
Hintergrundbild des notifiers der ein aktives userlogin Event anzeigt.
- **userlogin.ini**
Konfigurationsdatei des UserLogin notifiers.

Anzupassende Elemente: kioskclient

Zum Kioskclient siehe auch: Abschnitt [9.15](#)

Die Headerleiste des Hauptfensters (1) ist Kunden spezifisch anpassbar. Dabei spielen zwei Dateien eine Rolle:

- **opsiclientkiosk.png**
- **opsiclientkiosk.ini**

Die **opsiclientkiosk.png** enthält das Bild welches in diesen Bereich geladen wird.

Die **opsiclientkiosk.ini** definiert den Text und dessen Darstellung die in diesem Bereich angezeigt wird.

Beispiel:

```
[TitleLabel]
Text= Opsi Client Kiosk
FontName = Arial
FontSize = 15
FontColor = clWhite
FontBold = false
FontItalic = false
FontUnderline = false

[Tile]
Width = 220
Height = 200
Color = clCream
FontName = Arial
FontSize = 12
FontColor = clBlack
FontBold = false
FontItalic = false
FontUnderline = false
```

```
[TileRadio]
Fontsize = 10
None_color = clBlack
Uninstall_color = clRed
Setup_color = clGreen
```

Templates für diese Dateien finden Sie unter `/var/lib/opsi/depot/opsi-client-agent/files/opsi/opsiclientkiosk/opsi-client-agent` bzw. `C:\Program Files (x86)\opsi.org\opsi-client-agent\opsiclientkiosk\opsiclientkioskskin`

Das jeweils verwendete Icon kann durch Ablegen einer `kiosk.ico` Datei unter `/var/lib/opsi/depot/opsi-client-agent/files/opsi/custom/opsiclientkioskskin/` verändert werden.

Schutz Ihrer Änderungen vor Updates: Das custom Verzeichnis

(Seit opsi-client-agent Version 4.0.3.1)

Möchten Sie Änderungen, welche Sie an den oben genannten Dateien durchgeführt haben, davor schützen, dass selbige beim Einspielen einer neuen Version des opsi-client-agenten verloren gehen, so können Sie hierfür das `custom` Verzeichnis (`/var/lib/opsi/depot/opsi-client-agent/files/opsi/custom`) verwenden. Das komplette `custom` Verzeichnis wird bei der Installation einer neuen Version des opsi-client-agenten gesichert und wieder hergestellt, so dass hier gemachte Änderungen bei einem Update nicht verloren gehen.

- `custom/config.ini`
Diese custom Config-Datei wird als letzte eingelesen, somit gelten die in dieser Datei enthaltenen Werte statt den entsprechenden Einstellungen aus der Standard-Datei `cfg/config.ini`. Ausnahme: die Werte für `pckey` und `bootmode` werden nie aus dieser Datei geholt. Schreiben Sie in diese Datei **nur** die Werte, die Sie tatsächlich gegenüber dem Default geändert haben wollen.
- `custom/winstskin/*.*`
Alle Dateien aus diesem Verzeichnis werden bei der Installation des opsi-client-agent auf dem Client nach `C:\Program Files (x86)\opsi.org\opsi-client-agent\custom\winstskin` kopiert. Falls vorhanden, wird (ab opsi-winst Version 4.11.3.4) dieses `winstskin` Verzeichnis bevorzugt verwendet. Es muss alle benötigten Dateien und Einträge vollständig enthalten, da die Inhalte des Standard-`winstskin`-Verzeichnisses bei Verwendung des `custom/winstskin/` ignoriert werden.
- `custom/notifier/*.*`
Alle Dateien aus diesem Verzeichnis werden bei der Installation des opsi-client-agent auf dem Client nach `C:\Program Files (x86)\opsi.org\opsi-client-agent\notifier` kopiert und überschreiben dabei die entsprechenden aus dem serverseitigen Standard-Verzeichnis `files/opsi/dist/notifier/` stammenden Dateien.
- `custom/opsiclientd.conf`
wird, falls vorhanden, bei der Installation des opsi-client-agent auf dem Client nach `C:\Program Files (x86)\opsi.org\opsi-client-agent\opsiclientd` kopiert und überschreibt dabei die aus dem serverseitigen `files/opsi/dist/opsiclientd/` Verzeichnis stammende `opsiclientd.conf`. Es werden also nicht einzelne Werte überschrieben, sondern die komplette Datei, die deshalb alle benötigten Einstellungen vollständig enthalten muss.
Achtung:
Die Anpassung der `opsiclientd.conf` über diese Methode wird nicht empfohlen. Verwenden Sie zur Konfiguration Ihrer Clients Hostparameter/Configs wie im Kapitel zum opsi-client-agent beschrieben. Die Verwendung einer `custom/opsiclientd.conf` ist nur bei extrem komplexen Anpassungen der `opsiclientd.conf` sinnvoll. Wenn Sie diese Methode anwenden, müssen Sie bei jedem Einspielen einer neuen opsi-client-agent Version auf dem Server überprüfen, ob in der Default Datei `files/opsi/dist/opsiclientd/opsiclientd.conf` Änderungen gemacht oder neue Einträge hinzugefügt wurden, welche Sie in Ihrer Version nachpflegen müssen. Also:
Finger weg, es sei denn Sie wissen wirklich was Sie tun !
- `custom/opsiclientkioskskin/*.*` Alle Dateien aus diesem Verzeichnis werden bei der Installation des opsi-client-agent auf dem Client nach `C:\Program Files (x86)\opsi.org\opsi-client-agent\custom\opsiclientkioskskin` kopiert. Falls vorhanden, wird dieses `opsiclientkioskskin` Verzeichnis bevorzugt verwendet.

Ein nachträgliches Rechte nachziehen hilft Folgefehler zu vermeiden:

```
opsi-setup --set-rights /var/lib/opsi/depot/opsi-client-agent
```

6.1.5 Sperrung des Anwender Logins mittels opsi-Loginblocker

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

Ob der *opsi-Loginblocker* während der *opsi-client-agent*-Installation installiert bzw. aktiviert wird, kann über das *Product-Property* `loginblockerstart` konfiguriert werden.

opsi-Loginblocker unter NT5 (Win2k/WinXP)

Der opsi-Loginblocker (`opsigina.dll`) ist als *GINA* realisiert. Die *opsigina* wartet bis zum Abschluss der *Produktaktionen* oder dem Timeout (standard-Wert: 120 Sekunden) bei nicht erreichbarem *opsiclientd*. Danach wird die Kontrolle an die nächste *GINA* übergeben (in der Regel an die `msgina.dll`).

GINA steht hierbei für „Graphical Identification and Authentication“ und stellt die seitens Microsofts offiziell unterstützte Möglichkeit dar, in den Login-Prozess von Windows einzugreifen.

Gelegentlich ist es der Fall, dass bereits andere Softwareprodukte (z.B. Client für Novell-Netzwerke) eine *GINA* auf dem System installiert haben und empfindlich auf Eingriffe reagieren.

Generell sind mehrere ,nacheinander aufgerufene *GINAs* (*GINA-chaining*) durchaus möglich. Auch die *opsigina.dll* des opsi-Loginblocker ist für das genannte *GINA-chaining* vorbereitet. Sollte der beschriebene Fall bei Ihren Clients eintreten, informieren Sie sich bitte auf dem freien Supportforum (<https://forum.opsi.org>) nach bestehenden Anpassungsmöglichkeiten oder kontaktieren Sie die Firma *uib*.

opsi-Loginblocker unter NT6 (Vista/Win7)

Der *opsi-Loginblocker* für NT6 (Vista/Win7) ist als *credential provider filter* realisiert (*OpsiLoginBlocker.dll*). Er blockiert alle *credential provider* bis zum Abschluss der *Produktaktionen* oder dem Timeout (Standard-Wert: 120 Sekunden) bei nicht erreichbarem *opsiclientd*.

6.1.6 Nachträgliche Installation des opsi-client-agents

Die Anleitung zur nachträglichen Installation des *opsi-client-agents* finden Sie im Handbuch *opsi-getting-started* im Kapitel *Erste Schritte*.

Siehe aus Kapitel *opsi Software On Demand (Kiosk-Mode)*: Abschnitt 9.15

Installation des opsi-client-agent in einem Master-Image oder als Exe

Um den opsi-client-agent über ein versiegeltes (sysprep) Masterimage zu verteilen, muß der opsi-client-agent sich beim *aufwachen* also beim personalisieren des erstellten Klons auch neu personalisieren.

Dies geschieht am besten über eine Neuinstallation.

Dazu gehe Sie wie folgt vor:

- Kopieren Sie vom opsi_depot share den Inhalt des Verzeichnisses opsi-client-agent in ein temporäres Verzeichnis auf dem Master.
- Editieren Sie die Datei `files\opsi\cfg\config.ini` :
 - In der Sektion `[installation]` setzen Sie für `service_user=` einen User der Mitglied in der Gruppe opsi-admin ist.

- In der Sektion [installation] (nicht empfohlen) setzen Sie für `service_password=` das Klartextpasswort des users. Besser:
- In der Sektion [installation] setzen Sie für `service_hidden_password=` das base64 encodete passwort des users. Das encoden des Klartextpasswortes können sie über die entsprechende opsi-winst Funktion `base64EncodeStr(<string>)` durchführen oder z.B. über <http://www.base64encode.org/>
Wenn `service_hidden_password=` einen Wert hat wird `service_password=` ignoriert.
- In der Sektion [opsiclientd] setzen Sie für `config_service.url` = die Webserviceadresse Ihres opsi-config-servers z.B. `https://192.168.1.10:4447/rpc`
- Sorgen Sie dafür das in der Personalisierungsphase die Datei `silent_setup.cmd` aus dem temporären Verzeichnis aufgerufen wird.
- Nach dem Abschluß des Aufrufs der `silent_setup.cmd` kann das temporäre Verzeichnis gelöscht werden.

Wenn man möchte, kann man das ganze jetzt in eine selbstextrahierende .exe-Datei mit Programmaufruf verpacken, bspw. über das Programm filzip.

6.1.7 Das Systray Programm des opsi-client-agents

Das Systray Programm des opsi-client-agent erfüllt folgende Aufgaben:

- Regelmässige Information des Anwenders über anstehende Installationen (Optional)
- Information des Anwenders über anstehende Installationen auf Anforderung über das Kontextmenü
- Möglichkeit des Anwenders den Start der Installationen anzufordern.

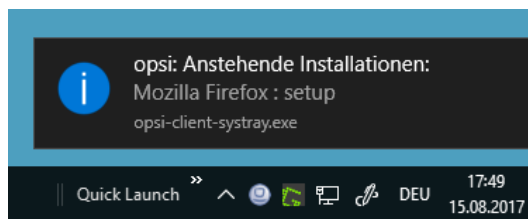


Abbildung 67: Message Fenster des opsi-systray Programms

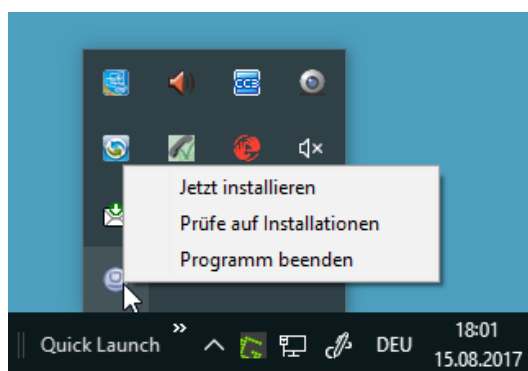


Abbildung 68: Kontext Menü (Rechte Maustaste) des opsi-systray Programms

Steuerung des opsi systray Programms über die Produktproperties des Produkts *opsi-client-agent*:

- **systray_install**
(true / false) Soll das opsi-systray Programm installiert werden ?
Default = false
- **systray_check_interval**
Alle wieviel Minuten soll das Programm überprüfen ob für den Client Installationen anstehen.
Default=180 (Kleine Werte hier, geben viel Last auf den Server)
Der Wert 0 bedeutet, dass keine regelmäßigen Prüfungen durchgeführt werden.
- **systray_request_notify_format**
Format der Benachrichtigung über anstehende Installationen.
Mögliche Werte:
"productid : request", "productname : request", "productname productversion : request"
default: "productname : request"

Logging des opsi systray Programms:

Das Programm loggt nach %Appdata%\opsi.org\log. D.h. in das Verzeichnis opsi.org\log im Anwendungsdatenverzeichnis des angemeldeten Users.

Zum Beispiel:

C:\Users\\AppData\Roaming\opsi.org\log\

Siehe auch Kapitel *opsi Software On Demand (Kiosk-Mode)*: Abschnitt [9.15](#)

6.2 Registryeinträge

6.2.1 Registryeinträge des opsiclientd

opsi.org/general

- **bootmode= <bkstd | reins>**
Beschreibt ob der Rechner gerade aus einer Reinstallation kommt.

opsi.org/shareinfo

- **depoturl**
<Url die zu den Softwarepaketen verweist. Muster: protokoll:\\server\share\dir>
Beispiel:
smb: \\opsi-server\opsi_depot
- **depotdrive**
<Laufwerksbuchstaben auf den depoturl gemountet wird>
Beispiel: P: (mit Doppelpunkt)

6.2.2 Registryeinträge des opsi-winst

opsi.org/winst

Diese Registry Einträge werden vom Winst selbst verwaltet und sollten nicht verändert werden.

```
"LastLogFilename"="C:\TMP\syslogin.log"
"ContinueLogFile"=dword:00000000
"RebootRequested"=dword:00000000
"SendLogToService"=dword:00000001
"NumberOfErrors"=dword:00000000
"ShutdownRequested"=dword:00000000
```

7 Security

7.1 Einführung

Opsi ist ein mächtiges Werkzeug zur zentralen Administration vieler Clients. Damit steht der *opsi-server* natürlich auch im besonderen Fokus der Sicherheitsbetrachtung. Wer den *opsi-server* kontrolliert, kontrolliert auch die Clients. Wer einem Client davon überzeugen kann er sei der richtige *opsi-server*, der kontrolliert den Client.

Wieviel Energie und Geld Sie in die Absicherung der opsi Infrastruktur stecken sollten hängt von Ihren Sicherheitsbedürfnis und vom Einsatzszenario ab. Ein *opsi-server* in der *cloud* ist z.B. gefährdeter als einer in einem abgeschlossenen Inselnetz.

Im folgenden haben wir die wichtigsten uns bekannten Maßnahmen und Probleme zusammengetragen.

Wir danken an dieser Stelle allen Kunden und Anwendern die uns auf Probleme hingewiesen und damit zur Sicherheit beigetragen haben. Wir bitten Sie darum Probleme die Ihnen auffallen zunächst an info@uib.de zu melden bevor das Problem öffentlich gemacht wird.

7.2 Informiert bleiben

Informationen über Security relevante opsi Updates werden veröffentlicht in:

News Bereich des opsi forums:

<https://forum.opsi.org/viewforum.php?f=1>

In der opsi Mailingliste:

https://lists.sourceforge.net/lists/listinfo/opsi-announce_de

7.3 Allgemeine Serversicherheit

Die opsi Software auf einem Server kann nicht sicherer als der Server selbst sein. Daher ist es wichtig, dass Sie Ihren *opsi-server* regelmäßig aktualisieren, d.h. die vom Distributor angebotenen Security Updates auch einspielen. Dies gilt sowohl für den *opsi-configserver* wie auch für die *opsi-depotserver*.

Sie können auf dem Server Programme installieren, welche Sie per E-Mail informieren wenn es Paketupdates gibt.

Debian, Ubuntu

apticron

RHEL, CentOS

yum-updatesd

Darüber hinaus gibt es viele Möglichkeiten Linuxserver weiter abzusichern. Dies ist aber nicht das Thema dieses Handbuchs. Gerne beraten wir Sie hierzu im Rahmen eines Support- und Pflegevertrages.

7.4 Authentifizierung des Clients beim Server

Der Client authentifiziert sich beim Server mit seinem FQDN sowie dem opsi-host-key. Client-seitig liegt dieser Key in der Datei `%programfiles%\opsi.org\opsi-client-agent\opsiclientd\opsiclientd.conf` und ist nur mit administrativen Rechten lesbar. Serverseitig liegen die opsi-host-keys im jeweiligen Backend (z.B. unter `/etc/opsi/pckey`).

Zusätzlich zu dieser Authentifizierung kann der `opsiconfd` angewiesen werden, zu überprüfen ob die IP-Nummer unter der sich der Client meldet auch zu dem angegebenen Clientnamen passt. Um dieses Verhalten zu aktivieren setzen sie in der `/etc/opsi/opsiconfd.conf` folgenden Parameter:

```
verify ip = yes
```

und reloaden den `opsiconfd`


```
service opsiconfd reload
```

**Achtung**

Verwenden Sie diese Feature nur wenn Sie eine vollständig funktionierende Namensauflösung (vorwärts wie rückwärts) für Ihre Clients haben.

7.5 Authentifizierung des Servers beim Client

Seit opsi 4.0.1 gibt es verschiedenen Möglichkeiten den Client prüfen zulassen ob der Server der kontaktiert vertrauenswürdig ist.

7.5.1 Variante 1: `verify_server_cert`

Bei der ersten Kontaktaufnahme zu einem opsi-server wird dessen SSL-Zertifikat akzeptiert und unter `C:\opsi.org\opsiclientd\server-certs` abgespeichert. Bei der nächsten Kontaktaufnahme wird der *public key* des gespeicherten Zertifikats ermittelt und mit diesem ein Zufallsstring sowie die eigenen Zugangsdaten verschlüsselt. Diese Daten werden an den Server übergeben. Der Server entschlüsselt mittels des *private key* seines SSL-Zertifikats diese Daten und sendet zum Beweis der erfolgreichen Entschlüsselung den Zufallsstring an den Client zurück. Nach erfolgreicher Prüfung des zurückgegebenen Zufallsstrings erachtet der Client den Server als vertrauenswürdig.

Auf diese Weise kann verhindert werden, dass per Angriff auf das DNS opsi-clients falsche Server als *opsi-server* akzeptieren. Diese Methode minimiert das Risiko, dass bei einem Serverumzug Clients den neuen Server nicht mehr akzeptieren, da sich die Serverzertifikate von einem auf einen anderen Server übertragen lassen. Außerdem ist keine Verteilung einer certification authority (CA) notwendig.

Die Schwäche dieser Methode ist die Möglichkeit eines *Man-in-the-middle* Angriffs, da aus den oben genannten Gründen das Zertifikat keiner weiteren Prüfung unterzogen wird.

Die Prüfung betrifft die Webservice Kommunikation mit dem *opsi-configserver*.

Wird im Rahmen der opsi WAN-Erweiterung für die WAN-Clients als `clientconfig.depot.protocol webdav` verwendet, so wird auch der *opsi-depotserver* entsprechend geprüft. Abschnitt [9.10.3](#)

Um diese Methode zu aktivieren, muss in der `opsiclientd.conf` in der Sektion `[global]` folgende Option gesetzt werden:

```
verify_server_cert = true
```

Mit dem folgenden Befehl richten Sie den Konfigurationseintrag ein, so das er vom *opsi-configed* bearbeitet werden kann:

```
opsi-admin -d method config_createBool opsiclientd.global.verify_server_cert "verify_server_cert" false
```

Aktivieren können Sie dies nun in dem Sie im *opsi-configed* in der *Serverkonfiguration* oder in den *Hostparameter* einzelner Clients den Wert von *false* auf *true* setzen.

**Achtung**

Dies sollte nur gemacht werden, wenn man genau weiß, was man tut. Ansonsten ist die Gefahr groß, dass die Clients den Kontakt zum Server verweigern und somit abgehängt sind.

7.5.2 Variante 2: `verify_server_cert_by_ca`

Dieser Mechanismus funktioniert analog zu der Art und Weise wie Browser die SSL-Zertifikate von Webservern prüfen. SSL-Zertifikate werden klaglos akzeptieren wenn diese auf genau den FQDN (bzw. Wildcard) ausgestellt sind unter dem der Server angesprochen wird und das Zertifikat von einer dem Browser bekannten CA signiert ist.

Der opsi-clientd enthält eine nur für diesen Zweck erstellte Root-CA der *uib gmbh* und akzeptiert Zertifikate, die durch die CA der *uib gmbh* signiert sind. Ebenso wird überprüft ob der *commonName* mit dem FQDN des Servers übereinstimmt. Ist eine der beiden Bedingungen nicht erfüllt, verweigert der Client die Kommunikation.

Diese Methode ist sicherer als die oben beschriebene, erfordert aber den Erwerb von Zertifikaten über die *uib gmbh*. Die Bedingungen und Preise zum Erwerb solcher Zertifikate erfahren Sie auf der [Preisliste](#) der *uib gmbh*. Die Überschüsse aus dem Zertifikatsverkauf fließen in die Pflege der opsi-Security.

Um diese Methode zu aktivieren, muss in der `opsiclientd.conf` in der Sektion `[global]` folgender Parameter gesetzt werden:

```
verify_server_cert_by_ca = true
```

Mit dem folgenden Befehl richten Sie den Konfigurationseintrag ein, so das er vom *opsi-configed* bearbeitet werden kann:

```
opsi-admin -d method config_createBool opsiclientd.global.verify_server_cert_by_ca "verify_server_cert_by_ca" false
```

Aktivieren können Sie dies nun in dem Sie im *opsi-configed* in der *Serverkonfiguration* oder in den *Hostparameter* einzelner Clients den Wert von *false* auf *true* setzen.



Achtung

Dies sollte nur gemacht werden, wenn man genau weiß, was man tut. Ansonsten ist die Gefahr groß, dass die Clients den Kontakt zum Server verweigern und somit abgehängt sind.

7.6 Authentifizierung beim controlserver des Client

Der *opsiclientd* besitzt eine Webservice-Schnittstelle die es erlaubt dem *opsiclientd* Anweisungen von aussen zu erteilen (Abschnitt 6.1.3).

Für den Zugriff auf den Webservice wird eine Authentifizierung benötigt. Dies geschieht entweder mittels des lokalen Administrator-Accounts (ein leeres Passwort ist unzulässig) oder mittels leerem Benutzernamen und dem *opsi-host-Schlüssels* als Passwort.

7.7 Konfiguration eines Admin-Networks

Die Idee eines Admin-Networks ist es, administrative Zugriffe auf Server nicht aus dem allgemeinen Produktiv-Netz zu erlauben, sondern nur von einem speziellen und abgesicherten Netzbereich.

Zwar müssen alle *opsi-clients* Zugang zum opsi-webservice haben, diese dürfen aber nur eingeschränkt Daten abrufen und ändern. Ein administrativer Zugang zum Webservice setzt die Mitgliedschaft in der Gruppe *opsiadmin* voraus.

Über die Option `[global] admin networks` in der `/etc/opsi/opsiconfd.conf` kann der administrative Zugriff auf den *opsiconfd* auf Verbindungen von bestimmten Netzwerkadressen eingeschränkt werden.

Es können mehrere Netzwerkadressen durch Kommas getrennt angegeben werden.

Nicht-administrative Client-Verbindungen können auch aus anderen Netzwerken erfolgen.

Der default ist:

```
admin networks = 0.0.0.0/0
```

und erlaubt Zugriff von allen Netzen.

Eine Konfiguration wie z.B.

```
admin networks = 127.0.0.1/32, 10.1.1.0/24
```

würde administrative Zugriffe nur vom Server selbst und aus dem Netz *10.1.1.0/24* erlauben.

7.8 Der user pcpatch

Der user *pcpatch* dient in opsi 4 dem Mounten des depot-Shares (*opsi_depot*) durch den Client.

Ausnahme hier von sind die Produkte:

- *opsi-wim-capture* und *opsi-local-image-capture* welche als user *pcpatch* auch den share *opsi_depot_rw* mit Schreibrechten mounten.
- *opsi-clonezilla* welches den share *opsi_images* mit Schreibrechten mountet.

Das Passwort des Benutzers *pcpatch* wird in der Regel verschlüsselt abgelegt und auch nur verschlüsselt übertragen. Es existieren jedoch auch unter gewissen Umständen Möglichkeiten das Passwort in Erfahrung zu bringen. Um den Schaden der hierdurch entstehen kann zu minimieren empfehlen wir folgende Maßnahmen:

In der */etc/samba/smb.conf* in allen Share-Definitionen ausser *opsi_depot* dem user *pcpatch* den Zugriff verbieten über den Eintrag:

```
invalid users = root pcpatch
```

Alternativ

In der */etc/samba/smb.conf* dem User *pcpatch* auf Leserechte beschränken durch den Eintrag in der [global] Sektion:

```
read list = pcpatch
```



Warnung

Für die Produkte *opsi-wim-capture* und *opsi-local-image-capture* muß der share *opsi_depot_rw* für *pcpatch* beschreibbar sein. Für das Produkt *opsi-clonezilla* muß der share *opsi_images* beschreibbar sein.

Als weitere Maßnahme sollten Sie das Passwort des Users *pcpatch* öfters ändern. Da das Klartext-Passwort niemandem bekannt sein muss, kann es z.B. durch den regelmäßigen Aufruf (z.B. per cronjob) des folgenden Scriptes auf ein zufälliges Passwort setzen.

```
opsi-admin -d task setPcpatchPassword $(< /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c16)
```

Falls Sie keine Netboot-Produkte verwenden, welche eine Anmeldung am Server als Benutzer *pcpatch* benötigen, so können Sie zusätzlich die Anmeldung des Benutzer *pcpatch* am Server unterbinden. Dazu weisen Sie in der Datei */etc/passwd* dem Benutzer *pcpatch* die Shell */bin/false* zu. Produkte, welche eine solche Anmeldung benötigen sind bspw. *ntfs-write-image* bzw. *ntfs-restore-image*.

7.9 Webservice-Zugriffsbeschränkungen

In der Datei */etc/opsi/backendManager/ac1.conf* kann der Zugriff auf bestimmte Methoden und Attribute der zurückgegebenen Werte beschränkt werden.

Die Beschränkung greift auf die Basis-Methoden des Webservices, für welche eingeschränkt werden kann welche Benutzer oder Gruppen zugreifen dürfen sowie für welche Attribute der Zugriff erlaubt ist.

Der Zugriff sollte für eine Absicherung auf eine Freigabe der verwendeten Methoden beschränkt werden. Falls unklar ist welche Methoden verwendet werden, so kann dazu die Ausgabe des *opsiconfd* über die aufgerufenen Methoden herangezogen werden, welche im Falle eines Neustarts oder Stops des Dienstes in der Datei */var/log/opsi/opsiconfd/opsiconfd.log* ausgegeben werden.

Weitere Informationen zu den Methoden des Webservice sind unter Abschnitt [5.4.1](#) zu finden.

7.10 Root Passwort des bootimages ändern

Das root Passwort des bootimages ist per default *linux123*. Dies kann aus Gründen der Sicherheit geändert werden. Wie das geht ist beschrieben in: Abschnitt [8.2.1](#)

8 opsi Produkte

8.1 Localboot-Produkte: Automatische Softwareverteilung mit opsi

Als Localboot-Produkte werden alle Produkte bezeichnet die nach einem lokalen Boot des Rechners über den *opsi-client-agent* installiert werden. Dies im Gegensatz zu den weiter unten beschriebenen Netboot Produkten Abschnitt [8.2](#).

8.1.1 opsi Standardprodukte

Die folgenden Localboot Produkte gehören zur Grundausstattung von opsi.

opsi-client-agent

Der *opsi-client-agent* ist der Clientagent von opsi und weiter oben ausführlich beschrieben: siehe Kapitel Abschnitt [6.1](#).

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 *opsi-client-agent* bei jedem Start, ob auf dem Server eine andere Version des *opsi-winst* verfügbar ist und holt sich diese im Zweifelsfall.

javavm: Java Runtime Environment

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

opsi-configed

opsi Graphical Management Interface als Applikation Für Windows und Linux. Siehe auch Kapitel: Abschnitt [4](#)

jedit

Java basierter Editor mit Syntax Highlighting für *opsi-winst* Skripte.

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.

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-product-id myprod
```

Siehe auch opsi-getting-started Manual.

opsi-template-with-admin

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-product-id myprod
```

Siehe auch opsi-winst-manual / opsi-script-manual

Kapitel: *Skript für Installationen im Kontext eines lokalen Administrators*

shutdownwanted

Führt den Rechner herunter, wenn keine weiteren Aktionen mehr gesetzt sind.

opsi-script-test

Große Sammlung von opsi-script Selbsttests. Diese kann als Beispielsammlung für funktionierende Aufrufe von opsi-script Befehlen verwendet werden.

opsi-wim-capture

Siehe auch Kapitel: Abschnitt [9.4](#)

opsi-winpe

Produkt zur einfachen Erzeugung eine opsi-winpe Siehe auch opsi-getting-started Manual, Kapitel *Erstellen eines PE*.

opsi-uefi-netboot

Siehe auch Kapitel: Abschnitt [9.6](#)

opsi-set-win-uac

Setzt den UAC-Level via opsi.

opsi-setup-detector

Siehe auch Kapitel: Abschnitt [9.19](#)

opsi-logviewer

Text viewer mit Filter nach Loglevel und Events.

Für Windows und Linux.

- Das von uib erstellte Tool opsi-logviewer öffnet jetzt auch Dateien, die u.a. in den Archivformaten zip oder gzip komprimiert wurden. Damit können dann Logdateien, die platzsparend als Archiv zugeschickt werden, direkt betrachtet werden. (Enthält ein Archiv mehrere Dateien, werden die Inhalte aneinandergehängt.)
- Das Setup.Skript ist um eine Linux-Unterstützung erweitert, so dass der opsi-logviewer auch auf einem Linux-Client automatisiert installiert werden kann.
- Die neue before-Abhängigkeit zu javavm sichert die Funktion des Startaufrufs (weil javavm die javaw.exe ins Systemverzeichnis kopiert)

config-win10

Konfiguriert verschiedene Windows 10 Einstellungen wie z.B. Sperrbildschirm, Hibernationboot, Telemetrie und Update-Verhalten.

In der Version 4.0.7 kamen neue Deaktivierungsmöglichkeiten hinzu.

- **change_power_plan** energieeinstellungen ändern.
- **config-updates** erlaubt es die Quelle der Updates zu Ändern. Die Updates werden dann entweder direkt von Microsoft Servern, einem lokalen Peer-To-Peer Netz oder einem Peer-To-Peer Netz aus dem Internet geladen. Die Option *disable* ist derzeit über ein eigenes Property namens *disable_updates* ermöglicht.
- **defer_upgrade** verschiebt Updates und Upgrades. Updates können um vier Wochen und Upgrades um acht Monate verschoben werden. Hierbei ist zu beachten das sicherheitsrelevante Updates trotz *defer* Option installiert werden. Featureupdates hingegen werden nicht installiert.
- **disable_advertising_id** deaktiviert die sogenannte Advertising ID. Diese speichert Daten über den Browserverlauf um benutzerspezifische Werbung einzublenden.
- **disable_cortana** deaktiviert den Cortana Sprachassistenten. Dieser sammelt diverse Daten über Eingaben und überträgt diese Daten an Server von Microsoft.
- **disable_customer_experience** deaktiviert das Sammeln von Daten im Bezug auf Daten zur Anwendungsverwendung.
- **disable_defender** deaktiviert den mit Windows 10 mitgelieferten Antivirenschutz namens *Defender*
- **disable_fast_boot** deaktiviert den Schnellstart und sorgt damit dafür, dass das Standard opsi-event *gui_startup* sauber funktioniert.
- **disable_font_streaming** sorgt dafür, dass nicht auf dem System installierte Fonts über das Internet nachgeladen werden.
- **disable_handwriting_share** Eine Besonderheit ist die Verwendung von Windows 10 auf Tablet-PCs. Hierbei werden Daten über Handschriften gesammelt und an einen Microsoft Server gesendet.
- **disable_location_sensors** deaktiviert das Sammeln von Daten über die aktuelle Geoposition des Geräts.
- **disable_lock_screen** Sperrbildschirm deaktivieren.
- **disable_mac** deaktiviert einen Dienst, welcher Daten über derzeit eingeloggte User sammelt und zu Microsoft überträgt.
- **disable_mrt** deaktiviert die Verwendung des *Malware Removal Tool*, kurz MRT. Dieser Dienst scannt in regelmäßigen Abständen vorhandene Dateien auf der Festplatte des Rechners und vergleicht diese mit einer Liste von potentiell gefährlicher Software.

- **disable_onedrive_sync** deaktiviert die OneDrive Dateisynchronisation.
- **disable_sending_feedback** ermöglicht es die Übertragung von Daten an Microsoft bei Fehlverhalten von Anwendungen zu beeinflussen.
- **disable_smbv1** deaktiviere Protokoll smbv1.
- Mit **disable_telemetry** ist es möglich die Menge der gesammelten Daten zu limitieren. Standardmässig werden sehr viele Daten übertragen. Wenn das Property auf *true* gesetzt wird, wird Windows so eingestellt, das nur noch sicherheitsrelevante Daten übertragen werden. Dies ist die niedrigste Stufe. Diese Security Stufe kann nur in der Windows 10 Enterprise und LTSB Version eingestellt werden. In den anderen Versionen von Windows 10 wird die nächst niedrige Stufe angewendet, Basic.
- **disable_updates** kappt bei der Option *true* die Verbindung zu Update Informationsquellen. Somit werden keine Updates gefunden per Updater. *false* ermöglicht die Verbindung zu Updatequellen.
- **disable_wifi_sense** ermöglicht es den *Wifi Sense* genannten Dienst zu deaktivieren. Dieser Dienst ermöglicht es gespeicherte WLAN Konfigurationen mit Kontakten zu teilen.
- **flashplayer_autorun**: Es gibt in Verbindung von Windows 10 mit dem Adobe Flashplayer eine Sicherheitslücke. Es wird empfohlen das Autorun-Feature des Flashplayers zu deaktivieren. Mit *false* wird der Flashplayer nicht mehr gestartet
- **online_search**: Bei jeder Suche über die integrierte Suchleiste in der Taskleiste werden auch Online-Ergebnisse geliefert. *true* ermöglicht eine solche online Suche, *false* verweigert diese.
- **sync_settings**: Wenn man Windows 10 in Kombination mit einem Microsoft Account nutzt, ist es möglich seine Einstellungen mit dem aktuellen Microsoft Konto zu synchronisieren. Setzt man das property *sync_settings* auf *false* wird dies deaktiviert.

```
[ProductProperty]
type: bool
name: disable_fast_boot
description: Disable Fastboot for proper opsi startup
default: True

[ProductProperty]
type: bool
name: disable_lock_screen
default: True

[ProductProperty]
type: bool
name: disable_telemetry
description: Disable telemetry data transmission
default: True

[ProductProperty]
type: bool
name: disable_cortana
description: Disable Cortana assistant
default: True

[ProductProperty]
type: bool
name: disable_customer_experience
description: Disable customer experience program
default: True

[ProductProperty]
type: bool
name: disable_mrt
description: Disable Malicious Software Removal Tool
default: True
```

```
[ProductProperty]
type: unicode
name: config_updates
multivalue: False
editable: False
description: Set Windows-Update behavior
values: ["AllowPeerToPeer", "LocalPeerToPeer", "MicrosoftOnly"]
default: ["MicrosoftOnly"]
```

```
[ProductProperty]
type: bool
name: disable_mac
description: Disable Microsoft Account communication
default: False
```

```
[ProductProperty]
type: bool
name: disable_advertising_id
description: Disable Microsoft Advertising ID
default: False
```

```
[ProductProperty]
type: bool
name: disable_updates
description: Disable Windows Updates
default: False
```

```
[ProductProperty]
type: bool
name: disable_defender
description: Disable Microsoft Windows Defender
default: False
```

```
[ProductProperty]
type: bool
name: disable_wifi_sense
description: Disable Wi-Fi Sense
default: False
```

```
[ProductProperty]
type: bool
name: disable_sending_feedback
description: Disable sending feedback and diagnostics
default: False
```

```
[ProductProperty]
type: bool
name: disable_font_streaming
description: Disable font streaming of not installed fonts
default: False
```

```
[ProductProperty]
type: bool
name: defer_upgrade
description: Defer Windows 10 Upgrade
default: True
```

```
[ProductProperty]
type: bool
name: flashplayer_autorun
description: Adobe Flashplayer: allow autorun?
default: False
```

```
[ProductProperty]
type: bool
name: location_sensors
description: Disable location and sensor detection
```



```
default: True

[ProductProperty]
type: bool
name: online_search
description: Disable online search during file or command search
default: True

[ProductProperty]
type: bool
name: disable_handwrite_sharing
description: Tablet-PC: Disable sharing of handriting information
default: True

[ProductProperty]
type: bool
name: sync_settings
description: Sync settings with AccountID
default: False
```

config-winbase

Paket zum Customizing der Grundeinstellungen von Oberfläche, Explorer usw..

8.1.2 Beeinflussung der Installationsreihenfolge durch Prioritäten und Produktabhängigkeiten

Seit opsi 4.0 wird die Installationsreihenfolge vom opsi-server unter Berücksichtigung von Produktabhängigkeiten und Produktprioritäten berechnet.

- Produktabhängigkeiten
definieren Abhängigkeiten und notwendige Installationsreihenfolgen zwischen Produkt-Paketen. Typische Beispiel ist die Abhängigkeit von Java Programmen von der Java Laufzeitumgebung (javavm).
- Produktprioritäten
dienen dazu, bestimmte Pakete in der Installationsreihenfolge nach vorne oder nach hinten zu schieben. So ist es z.B. sinnvoll Servicepacks und Patches an den Anfang einer Installation zu legen und eine Softwareinventarisierung an das Ende.
Produktprioritäten sind Zahlen zwischen 100 und -100 (0 ist default)

Wie diese beiden Faktoren gegeneinander gewichtete werden sollen kann unterschiedlich gesehen werden. Daher stellt opsi zwei Algorithmen zur Verfügung.

Die Umstellung zwischen diesen Algorithmen erfolgt entweder:

im *opsi-configed*, in der Server-Konfiguration

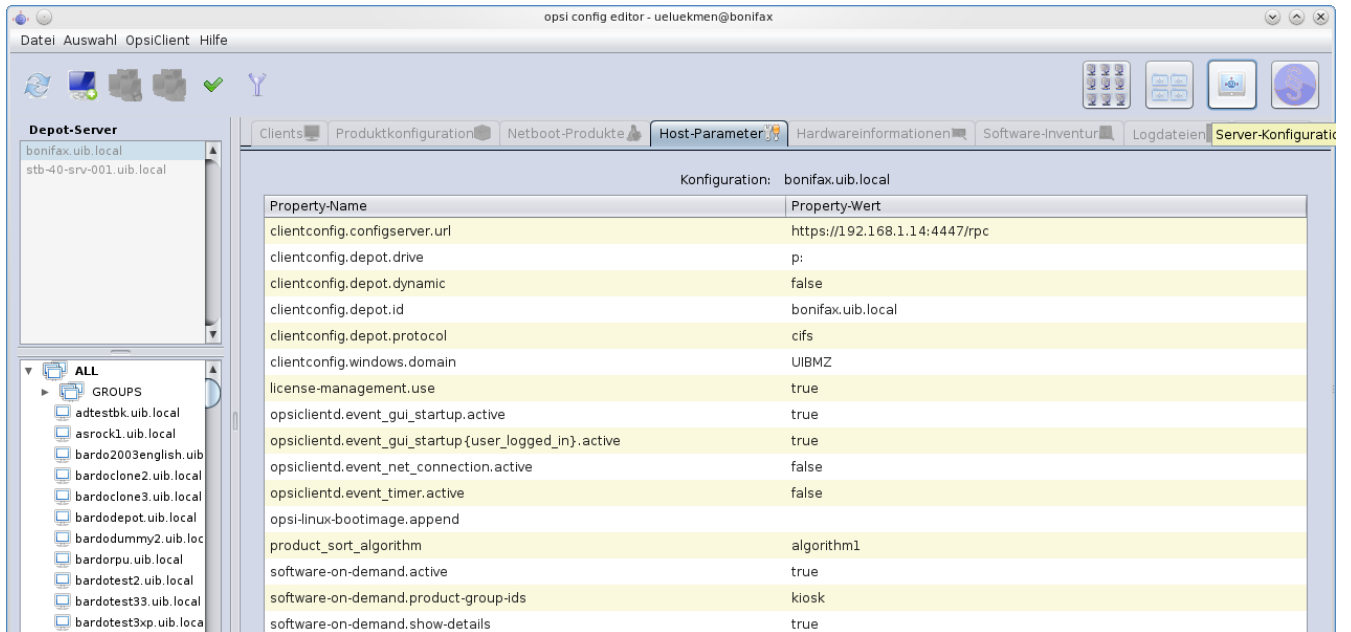


Abbildung 69: opsi-configed: Serverkonfiguration

oder auf der Kommandozeile mit folgendem Befehl:

```
opsi-setup --edit-config-defaults
```

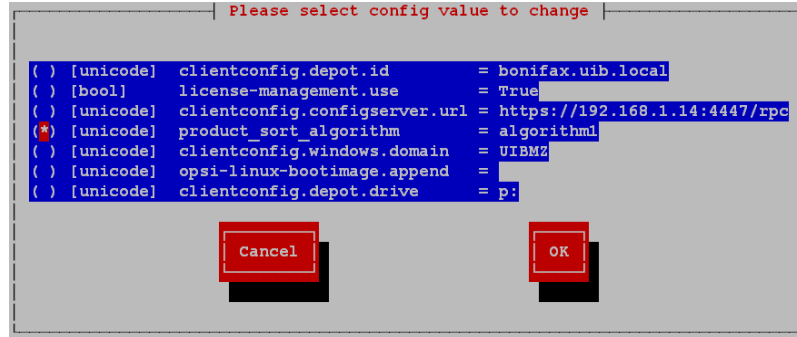


Abbildung 70: Wählen des Sortieralgorithmus: Teil 1

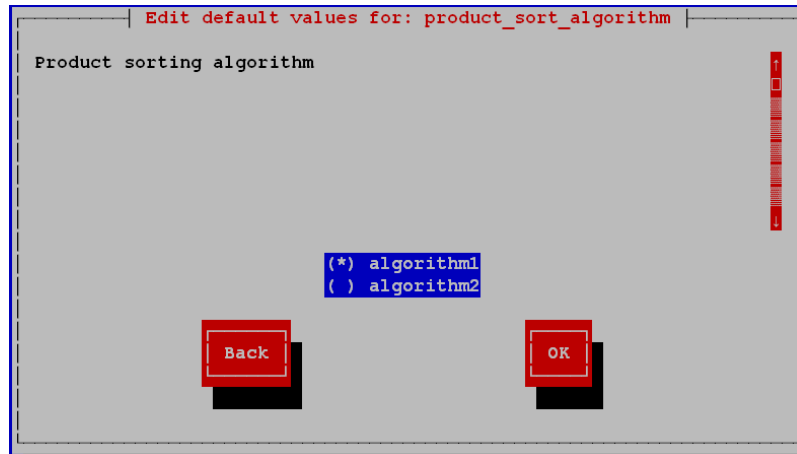


Abbildung 71: Wählen des Sortieralgorithmus: Teil 2

Algorithm1: Produktabhängigkeit vor Priorität (Default)

Bei diesem Algorithmus werden zunächst die Produkte anhand Ihrer Prioritäten sortiert und dann aufgrund der Produktabhängigkeiten nochmals umsortiert. Hierdurch kann natürlich ein Produkt mit sehr niedriger Priorität weit nach vorne geschoben werden weil es von einem anderen Produkt als *required before* benötigt wird. Auf der anderen Seite wird vermieden das es zu Installationsproblemen aufgrund nicht aufgelöster Produktabhängigkeiten kommt.

Der Algorithmus 1 sorgt dafür, das die Installationsreihenfolge konstant ist, unabhängig davon wieviele Produkte auf setup stehen. Diese Reihenfolge entspricht der Reihenfolge welche im configed angezeigt wird wenn die Produkte nach der Spalte Position sortiert werden.

Damit ist gesichert, dass bei einer mit "ExitWindows /immediateReboot" nur unterbrochenen Abarbeitung eines setup-Skripts nach dem Reboot direkt die Bearbeitung des unterbrochenen Skripts weitergeführt wird.

Algorithm2: Produktpriorität vor Abhängigkeit

Dieser Algorithmus geht von dem Gedanken aus, dass es in der Praxis im wesentlichen drei Prioritätsklassen gibt:

- Produkte, welche am Anfang installiert werden sollen wie z.B. OS-Patches und Treiber, durch die der PC in seinen Standard-Zustand gebracht wird. Wird realisiert durch Zuweisen einer hohen Priorität (maximal +100).
- "Normale" Produkte, die Anwendungen installieren (Default-Priorität 0).
- Produkte, die möglichst am Ende eingespielt werden sollen wie z.B. Softwareinventarisierung. Realisiert durch Zuweisen einer niedrigen Priorität (niedrigste mögliche -100).

Die Auflösung der Produktabhängigkeiten erfolgt nur innerhalb einer Prioritätsklasse. Hierdurch ist sichergestellt, dass Produkte mit einer hohen Priorität auch tatsächlich am Anfang installiert werden. Prioritätsklassen übergreifende Produktabhängigkeiten werden nicht berücksichtigt bzw. führen zu einer Warnung. Daher ist beim Packen zu beachten, dass Produktabhängigkeiten nur innerhalb einer Prioritätsklasse definiert werden.

Die Produktabhängigkeiten werden hier so interpretiert, dass sie bei "normalen" Produkten automatisch zu einer konsistenten, alle Abhängigkeiten berücksichtigenden Reihenfolge führen. Wurden widersprüchliche (zirkuläre) Abhängigkeiten definiert, wird ein Fehler angezeigt.

Bei den für die PC-Einrichtung grundlegenden Produkten mit hohen Prioritäten wird dagegen der Administrator – ähnlich wie etwa bei Unix-Startskripten – die genaue Reihenfolge von Hand festlegen, indem er pro Produkt entsprechend der gewünschten Reihenfolge je eine spezifische Priorität zwischen +100 und +1 setzt. Ähnliches gilt für die finalen Produkte mit niedrigen Prioritäten.

Erstellung von Prioritäten und Produktabhängigkeiten

Prioritäten und Produktabhängigkeiten gehören zu den Meta-Daten eines Produktes. Diese werden bei der Erstellung eines Produktes mit dem Befehl `opsi-newprod` abgefragt.

Diese Metadaten werden im control file des Produktes abgelegt und können dort editiert werden. Nach einer Veränderung im control file muss das Produkt neu gepackt und installiert werden.

Siehe hierzu auch das Kapitel *Erstellen eines opsi-Product-Paketes* im opsi-getting-started Handbuch.

8.1.3 Einbindung eigener Software in die Softwareverteilung von opsi

Die Anleitung zur Einbindung eigener Software finden Sie im Handbuch opsi-getting-started.

8.2 Netboot Produkte

8.2.1 Parametrisierung vom Linux Installationsbootimage

Das opsi-linux-bootimage hat eine Reihe von Standardeinstellungen, welche das Verhalten beeinflussen. Sollte nach dem Laden des Bootimages der Bildschirm schwarz bleiben oder die Netzwerkkarte nicht (korrekt) funktionieren, so muss evtl. für diese Hardware die Startparameter des Bootimages angepasst werden.

Dies können Sie im *opsi-configed* im Tab *Hostparameter* am Eintrag *opsi-linux-bootimage.append* tun.

Typische Werte sind hier (einzeln oder kombiniert):

- `acpi=off`
- `noapic`
- `irqpoll`
- `reboot=bios`

Eine weitere wichtige Standardeinstellung ist das Passwort von root im Bootimage. Dieses ist per default *linux123* und sollte aus Sicherheitsgründen ebenfalls auf diesem Weg abgeändert werden.

Um diese angesprochenen Modifikationen durch zu führen, muss man die Konfiguration: *opsi-linux-bootimage.append* am besten in der Serverkonfiguration anpassen.

Die hier wichtige Option ist *pwsh*. Dieser Option muss man das verschlüsselte Passwort als Hash-Wert mitgeben. Dieser wird dann automatisch beim Booten in die Datei */etc/shadow* geladen. Somit wird bevor ein Login auf dem Client möglich ist, das Passwort verändert. Um diesen Hash zu bekommen gibt es verschiedene Wege. Um sicher zu gehen, dass man den richtigen Hash in der richtigen Formatierung und Art (MD5, SHA1, etc...) bekommt, empfehlen wir folgendes Vorgehen.

Möchte man ein Python Skript vor dem eigentlichen Netboot Produkt starten gibt es noch die Parameter:

- `pre-execute`
- `pre-script`

Als zusatz benötigen die Parameter noch eine Adresse, von der das entsprechende Skript gezogen werden soll. Dies kann eine *http://* oder *tftp://* Adresse sein. Hier ein Beispiel:

- `tftp://172.16.166/linux/test.py`

Es ist zu beachten das der Port für die Kommunikation mit dem tftp-Server standardmäßig auf 69 gestellt ist.

Einen opsi-Client per PXE oder mit der aktuellen opsi-client-boot-cd starten. Dann per Putty oder direkt vom *opsi-server* aus per ssh als root eine Verbindung aufbauen. Vom *opsi-server* aus:

```
ssh root@<client.domain.tld>
```

Das Passwort lautet linux123. Dann einfach das Passwort von root neu setzen:

```
passwd
```

Nun muss man den gesetzten Hash holen.

```
grep root /etc/shadow
```

Die Ausgabe sollte nun folgendermaßen aussehen:

```
root:$6$344YXKIT$D4RPZfHMmv8e1/i5nNkOfaRN2oYNobCEjCHnkehiEFA7NdkDW9KF4960HBmyHHqOkD2FBLHZoTdr5YoD1IoWz\  
/:14803:0:99999:7:::
```

Um das Passwort zu setzen, reicht es wenn man den Hash kopiert, der nach dem ersten Doppelpunkt beginnt und beim zweiten Doppelpunkt in der Zeile aufhört. Die daraus resultierende Option für die Konfiguration *opsi-linux-bootimage.append* wäre:

```
pwh=$6$344YXKIT$D4RPZfHMmv8e1/i5nNkOfaRN2oYNobCEjCHnkehiEFA7NdkDW9KF4960HBmyHHqOkD2FBLHZoTdr5YoD1IoWz/
```

8.2.2 Automatische Betriebssysteminstallation unattended

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-client-boot-cd* 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 Konfigurationsinformationen 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-client-agent* zur Einbindung der automatischen Softwareverteilung installiert.
- Die automatische Softwareverteilung installiert die gesamte Software, die gemäß Konfigurationsdatei auf diesen Rechner gehört.

Voraussetzungen

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 auch von der *opsi-client-boot-cd* das Bootimage gebootet werden.

Das opsi-Installationspaket für das zu installierende Betriebssystem muss auf dem opsiserver installiert sein. In den folgenden Abschnitten wird als Beispiel jeweils Windows 10 angenommen.

PC-Client bootet vom Netz

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

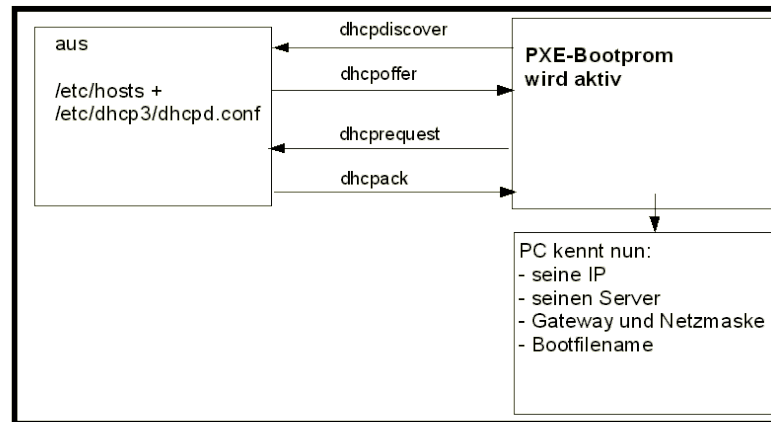


Abbildung 72: Schritt 1 beim PXE-Boot

Der PC kennt zu Beginn lediglich seine Hardware-Adresse (= hardware ethernet, MACNummer 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 PCs 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.

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 `x/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 `localboot 0` Lädt der PC diese Datei, findet also keine Installation statt, sondern das lokal installierte Betriebssystem wird gestartet.

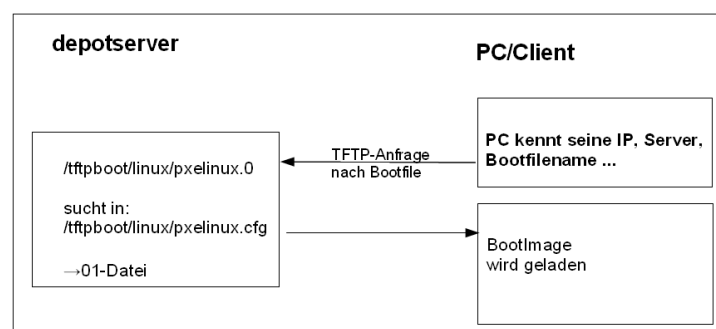


Abbildung 73: Schritt 2 beim PXE-Boot

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 `opsipxeconfd`.

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" (initiale root disc) Filesystem (`/tftpboot/linux/miniroot.bz2`).

Das Bootimage, das nun geladen wird, ist Linux basiert.

PC-Client bootet von CD

Analog zu dem Bootvorgang per `tftp` mit Hilfe des PXE-bootproms kann das Bootimage auch direkt von der `opsi-client-boot-cd` 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.

Lesen Sie hierzu auch das Kapitel *Anlegen eines neuen opsi-Clients mit Hilfe der opsi-client-bootcd* in Getting-Started Handbuch.

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.

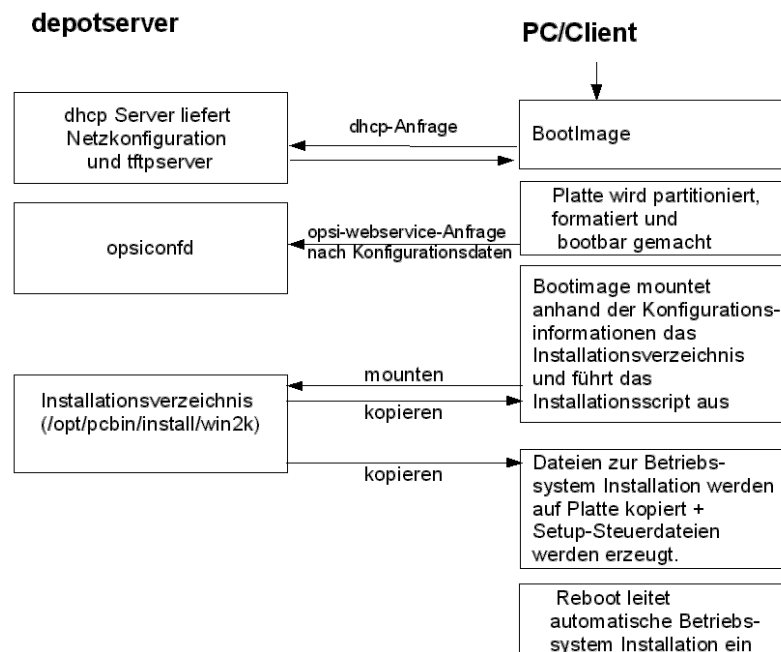


Abbildung 74: Ueber PXE-Boot geladenes Bootimage bereitet Festplatte zur Betriebssysteminstallation vor

Ergänzt wird dieses Informationspaket durch Angaben aus der dhcp-Antwort (z.B. wer ist der tftp-Server) sowie mit über den Webservice ermittelte Informationen. Die gesammelten Informationen werden für die Weiterverarbeitung durch das eigentliche Installationskript 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 Installationskript 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-10-Installation skizziert.

Vorbereitung der Festplatte: Auf der Platte wird eine 4 GB große FAT32 Partition angelegt, formatiert und bootfähig gemacht. Sofern nicht anders angegeben, wird der Rest der Festplatte mit einem NTFS Dateisystem formatiert. Auf diese Partition wird das Betriebssystem installiert.

Kopieren der Installationsdateien: Die Dateien für die Installation des Betriebssystems werden von dem Installationsshare des Servers (z.B. /var/lib/opsi/depot/win10/installfiles) auf die lokale Platte kopiert. Das Gleiche gilt für das Setup-Programm des 'opsi-client-agent's zur Einrichtung der automatischen Softwareverteilung auf dem PC.

Einpfelegen der Konfigurationsinformationen: Unter den 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). Ein Beispiel für eine zu patchende Datei ist die *unattend.xml*. Sie steuert das „unbeaufsichtigte“ Installieren von Windows.

Reboot vorbereiten: Der *Bootloader* wird 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.xml* als Steuerdatei zu verwenden.

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.

Die Installation von Betriebssystem und opsi-client-agent

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 der Installations-Partition 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-client-agent's auszuführen 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, für welche der Installationsstatus *installed* oder die angeforderte Aktion *setup/update* ist, 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, werden im Standard-Bootmodus nur Pakete installiert, bei denen die Angeforderte Aktion *setup/update* ist.) Damit ist der PC fertig installiert.

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.xml* 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. Die zu patchenden Parameter werden aus den Properties des jeweiligen Netboot-Produkts gelesen und in eine Variable im Bootimage geschrieben.

```
Usage: patcha [-h|-v] [-f <params file>] <patch file>

Fill placeholders in file <patch file>
Options:
-v Show version information and exit
-h Show this help
-f <params file> File containig key value pairs
If option not given key value pairs from kernel cmdline are used
```

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

Aufbau der Produkte zur unattended Installation

Die Informationen zum *Aufbau der Produkte zur unattended Installation* finden Sie im Handbuch opsi-getting-started.

Vereinfachte Treiberintegration in die automatische Windowsinstallation

Die Informationen zum *Vereinfachte Treiberintegration in die automatische Windowsinstallation* finden Sie im Handbuch opsi-getting-started.

8.2.3 Hinweise zu den NT6 Netbootprodukten (Win 7 bis Win 10)

Voraussetzungen Alle Netbootprodukte der Version $\geq 4.1.0.0$ benötigen einen auf dem Server installierten opsi-winst $\geq 4.12.0.13$. Diese Netbootprodukte laufen auch unter opsi 4.0.x.

Multidiskmode Der neue Multidiskmode bietet eine Unterstützung der Betriebssystem Installation auf Systemen mit mehreren Festplatten. Dabei kann gezielt die gewünschte Zielfestplatte ausgewählt werden. Es kann auch gezielt *die erste SSD* oder *die erste rotierende Festplatte* ausgewählt werden.

Der Multidiskmode ist nur implementiert für die Property-Einstellung: **winpenetworkmode = true**.



Wichtig

Wenn Sie auf einem Rechner mit **MBR BIOS** über das Property `multidiskmode` eine Platte wählen, so müssen Sie dafür sorgen, dass diese Platte auch die erste Platte in der BIOS Bootreihenfolge ist.

Bei **UEFI BIOS** Systemen müssen Sie nichts unternehmen, da hier die Bootreihenfolge durch die Installation gesteuert werden kann.

Verhalten im PE Bei der Windows Betriebssystem Installation wird durch das opsi-linux-bootimage die Festplatte vorbereitet und ein Win-PE Partition erstellt. Diese wird gebootet um die eigentliche Windowsinstallation zu starten. In den 4.1.0.0 Produkten wird hier opsi-script gestartet um die Notwendigen Arbeiten durch zu führen. Die Vorteile dieses Vorgehens sind:

- Einfacheres und übersichtlicheres Scripting
- Erstellung einer Log-Datei der Vorgänge im Win-PE
- Automatische Übertragung der Logdatei an den opsi-server.
(Diese Logdatei wird an das bootimage Log angehängt.)

Die Netbootprodukte zur Installation von NT6 Betriebssystemen enthalten eine Fülle von Produktproperties, welche in Ihrer Funktion in der Folge erläutert werden sollen:

Property-Name	Property-Wert
additional_drivers	
administrator_password	*****
askbeforeinst	false
boot_partition_label	BOOT
boot_partition_letter	-
boot_partition_size	0
data_partition_create	true
data_partition_label	DATA
data_partition_letter	D
data_partition_preserve	never
fullname	Name
imagename	Windows 10 Pro N
installto	disk
multi_disk_mode	0
orgname	Orgname
pre_format_system_partitions	true
preserve_winpe_partition	false
productkey	
setup_after_install	
system_keyboard_layout	0407:00000407
system_language	de-DE
system_timezone	W. Europe Standard Time
use_raid1	false
windows_partition_label	WINDOWS
windows_partition_size	100%
winpe_dir	auto
winpe_inputlocale	0407:00000407
winpe_partition_size	4000M
winpe_uilanguage	de-DE
winpe_uilanguage_fallback	de-DE
winpenetworkmode	true

Abbildung 75: NT6 Productproperties

additional_drivers

Liste von Verzeichnissen unterhalb von `<productid>\drivers\drivers\additional`. Alle Treiberverzeichnisse unterhalb der angegebenen Verzeichnisse werden unabhängig von der automatischen Treibererkennung zusätzlich in die Installation mit eingebunden. Wird hierüber Treiber für ein Gerät eingebunden, so wird für dieses Gerät kein weiterer Treiber über die automatische Treiberintegration mehr eingebunden.

administrator_password

Hier kann das Passwort angegeben werden, welches bei der Installation für den lokalen Administrator gesetzt wird.
Default = *nt123*

askbeforeinst

Soll vor Beginn der Installation gefragt werden

boot_partition_label

Label der *boot_partition* (Bitlocker Partition)

boot_partition_letter

Laufwerksbuchstabe der *boot_partition* (Bitlocker Partition)

boot_partition_size

Größe der *boot_partition* (Bitlocker Partition). 0 = keine Erstellen

data_partition_label

Label der Datenpartition, wenn eine erstellt wird.

data_partition_letter

Laufwerksbuchstabe der Datenpartition, wenn eine erstellt wird.

data_partition_preserve

Soll die Datenpartition bei einer Reinstallation erhalten bleiben

fullname

Vollständiger Name des Lizenznehmers wie er der Installation übergeben wird.

imagename

Name der Variante des Betriebssystems das zu installieren ist.

installto

Diese Property ist nicht editierbar. Es dient beim Ablauf zur Unterscheidung zwischen standard (disk), opsi-local-image (oli) und opsi-vhd (vhd).

Bitte: Finger weg.

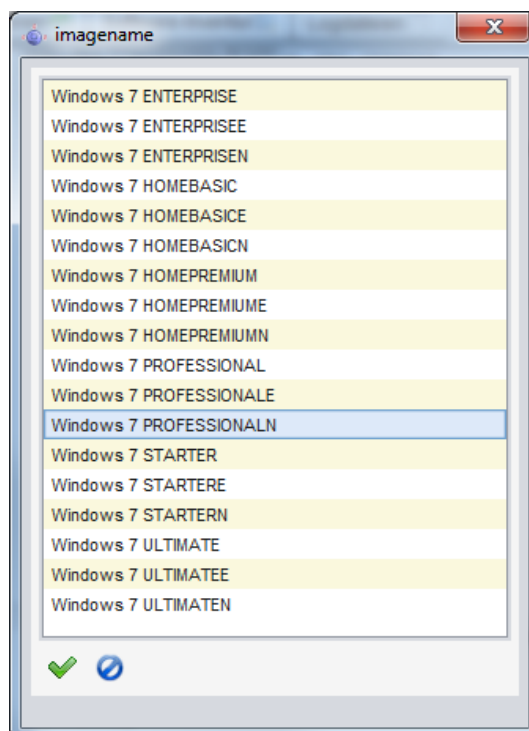


Abbildung 76: NT6 Imagenames

multi_disk_mode

Diese Property dient zur Wahl der Festplatte auf die installiert werden soll.

Mögliche Werte sind: "0","1","2","3","prefer_ssd","prefer_rotational"

Die Werte "0","1","2","3" geben den Index der Festplatte direkt an ("0"= 1. Festplatte)

Der Wert "prefer_ssd" wählt die erste SSD Platte aus.

Der Wert "prefer_rotational" wählt die erste klassische Platte (mit rotierenden Scheiben) aus.

Das Property wird auf Systemen mit nur einer Platte ignoriert.

Default = "0"

orgname

Vollständiger Name der Firma / Organisation des Lizenznehmers wie er der Installation übergeben wird.

pre_format_system_partitions

Sollen vorherige Partitionen formatiert werden um Spuren vorheriger Installationen zu löschen? (benötigt Zeit)

preserve_winpe_partition

Soll die WinPE Partition nach der OS-Installation erhalten bleiben

productkey

Lizenzschlüssel zur Installation. Wird nur ausgewertet wenn der Hostparameter `license-management.use` auf `false` steht. Ansonsten wird der Lizenzschlüssel aus dem Lizenzmanagement geholt.

setup_after_install

Welches Produkt soll nach dem Betriebssystem installiert werden?

system_keyboard_layout

Sprachauswahl für die Tastatur. (siehe: [link:http://msdn.microsoft.com/en-us/goglobal/bb895996](http://msdn.microsoft.com/en-us/goglobal/bb895996))

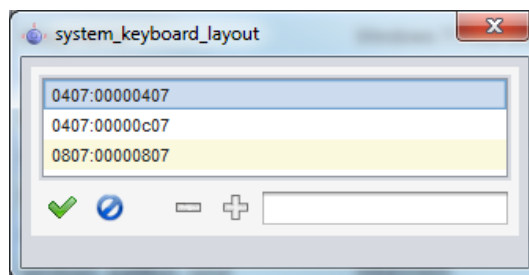


Abbildung 77: Sprachauswahl für die Tastatur

system_language

Sprachauswahl für das System.

system_timezone

Zeitzoneneinstellung

use_raid1

Soll ein RAID1 (SOFTWARE) angelegt werden?

windows_partition_label

Label der Partition (Festplatte C:) auf die das Betriebssystem installiert werden soll.

windows_partition_size

Größe der Partition (Festplatte C:) auf die das Betriebssystem installiert werden soll. Die Angabe kann in Prozent der Festplatte oder in absoluten Zahlen (G=Gigabyte) erfolgen. Wird ein anderer Wert als 100% gewählt, so wird auf dem verbleibenden Rest der Faspalte eine *data_partition* angelegt.

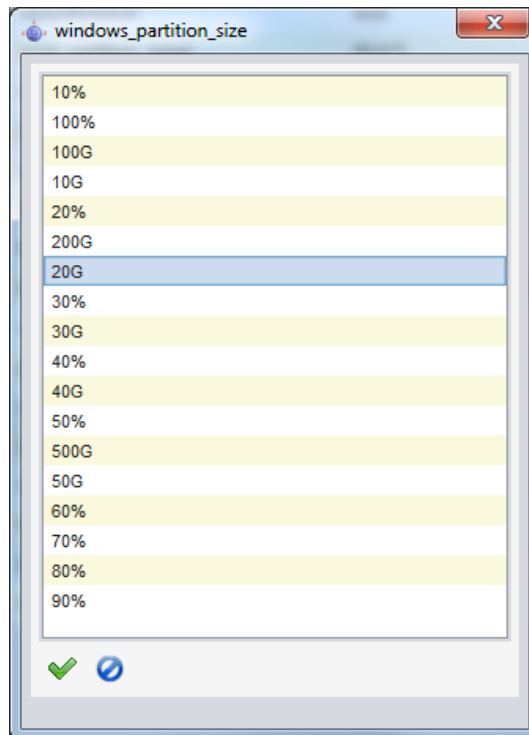


Abbildung 78: Größe der C: Partition

winpe_dir

Diese Property dient zu Debug Zwecken.

Der Wert "auto" wählt ermittelt das passende standard winpe Verzeichnis im Verzeichnis des Netbootproduktes.

Dies ist *winpe* bzw. *winpe_uefi*

Andere Werte müssen auf ein entsprechendes, existierendes Verzeichnis im Verzeichnis des Netbootproduktes verweisen.

Default = *auto*

winpe_inputlocale

Microsoft-Windows-International-Core-WinPE InputLocale

winpenetworkmode

Soll die Betriebssysteminstallation über den gemounteten Netzwerkshare vom PE aus erfolgen (true) oder sollen alle Installationsdateien vorher auf die Festplatte kopiert werden (false).

winpe_uilanguage

Microsoft-Windows-International-Core-WinPE

winpe_uilanguage_fallback

Microsoft-Windows-International-Core-WinPE

winpenetworkmode

Beim Wert true versucht lädt das WinPE die Installationsdaten vom opsi share. Beim Wert false werden die kompletten Installationsdaten auf die lokale Festplatte geladen und die Installation startet von lokaler Platte

8.2.4 memtest

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

8.2.5 hwinvent

Das Produkt *hwinvent* dient dazu eine Hardwareinventarisierung des Clients durchzuführen.

8.2.6 wipedisk

Das Produkt *wipedisk* überschreibt die gesamte Festplatte (`partition=0`) oder einzelne Partitionen mit unterschiedlichen Mustern. Die Anzahl der Schreibvorgänge wird über das Product-Property *iterations* gesteuert (1-25).

8.3 Inventarisierung

Zur Inventarisierung stehen die Localbootprodukte *hwaudit* und *swaudit* sowie das Netboot Produkt *hwinvent* zur Verfügung.

8.3.1 Hardware Inventarisierung

Die Hardwareinventarisierung ist unter opsi über eine Konfigurationsdatei gesteuert. Das bedeutet, dass die Information wie und welche Daten erhoben werden, nicht in den entsprechenden Produkten *hwaudit* und *hwinvent* fest verdrahtet sind. Vielmehr werden diese Produkte über eine Konfigurationsdatei gesteuert. Dazu wird die Konfigurationsdatei bei jeder Ausführung über den opsi Webservice eingelesen und interpretiert. Gleichzeitig steuert diese Konfigurationsdatei auch den Aufbau der Datenbank, so dass eine Erweiterung dieser Konfigurationsdatei auch eine Erweiterung der Datenhaltung nach sich zieht.

Die Konfigurationsdatei ist die `/etc/opsi/hwaudit/opsihwaudit.conf`.

In dieser Datei werden alle zu inventarisierenden Objekte definiert und beschrieben, wie die zu diesem Objekt gehörenden Daten zu erheben sind (unter Linux und unter Windows). Gleichzeitig wird darüber auch die dazu gehörige Datenstruktur definiert. Zur Vereinfachung enthält diese Konfigurationsdatei Vererbungsmechanismen die an eine Objektorientierung angelehnt sind. Hintergrund hierfür ist die Tatsache, dass viele Objekte identische Datenfelder wie z.B. *Name* und *Vendor* enthalten. Diese allgemeinen Informationen werden so in *virtual* Hardwareklassen definiert. Die eigentlichen Inventarisierungsobjekte sind dann *structural* Hardwareklassen, welche viele Eigenschaften von übergeordneten *virtual* Klassen erben können.

Zur Erläuterung dieses Mechanismus ein Beispiel:

So definiert die Konfigurationsdatei zunächst eine *virtual Class* Namens *"BASIC_INFO"*. Diese definiert die Eigenschaften (*Values*):

- "name"
- "description"

Als nächstes folgt die *virtual Class* Namens *"HARDWARE_DEVICE"* welche alle Eigenschaften von *"BASIC_INFO"* erbt und folgende zusätzliche definiert:

- "vendor"
- "model"
- "serialNumber"

Als nächstes folgt als erstes Objekt welche wir in der Inventarisierung auch finden, die erste *structural Class* Namens *"COMPUTER_SYSTEM"*, welche alle Eigenschaften von *"HARDWARE_DEVICE"* erbt und folgende zusätzliche definiert bzw. überschreibt:

- "name"
- "systemType"

- "totalPhysicalMemory"

Im Rahmen der Definition einer Klasse und ihrer *Values* werden verschiedene Eigenschaften beschrieben:

- Klassen definition:
 - "Type"
ist "STRUCTURAL" oder "VIRTUAL"
 - "Super"
gibt die Klasse an von der geerbt wird.
 - "Opsi"
gibt den Namen der Klasse an, der auch später in opsi als Anzeigenamen verwendet wird.

Weiterhin können in der Klassendefinition angegeben werden, wie diese Daten erhoben werden. Diese Informationen können aber auch bei der Definition der *Values* stehen.

- Für die Inventarisierung unter Linux:
 - "Linux": "[<command>]<parameter>"
Ausführung des Kommandozeilenprogramms <command> mit dem Argument <parameter>.
 - "Python": "<python code with place holder>"
Ausführung des angegeben Python codes wobei zunächst der Platzhalte durch die schon ermittelten Werte ersetzt wird.
- Für die Inventarisierung unter Windows:
 - "WMI": "<wmi select statement>"
auszuführende WMI Abfrage
 - "Cmd": "<Python text object with place holder>"
Der Platzhalter ist in diesem Fall der relative Pfad zu einem ausführbarem Programm, dessen Ausgabe den Platzhalter ersetzt.
 - "Registry": "[<registry key>] <value name>"
Aus der Registry wird in <registry key> der Wert von <value name> ausgelesen.
Das Auslesen der Registry erfolgt Architektur spezifisch. Das heißt, auf einem 64 Bit System wird der 64 Bit Zweig der Registry ausgelesen.
- Valuedefinition:
 - "Type": "<MySQL Datenbanktyp>"
<MySQL Datenbanktyp> gibt den Datentyp an in dem dieser Wert in der Datenbank angelegt wird.
 - "Scope": "<scope>"
das Feld <scope> wird folgendermaßen verwendet:
"g" bedeutet: Dieses Attribut ist bei allen Geräten dieses Typs gleich.
"i" bedeutet: Dieses Attribut kann bei Geräten dieses Typs unterschiedliche Werte haben.
 - "Opsi": "<id>"
dabei ist <id> der opsi interne Name des Feldes. Dieser kann zur Ausgabe über die Dateien in /etc/opsi/hwaudit/locales wiederum lokalisiert werden.
 - "WMI": "<id or command>"
dabei ist <id or command> entweder der Name unter dem der in der Klassen definition angegebene WMI Befehl den Wert ausgibt oder ein eigener WMI Befehl.
 - "Linux": "<id>"
dabei ist <id> der Name unter dem der in der Klassen definition angegebene Linux Befehl den Wert ausgibt.

- "Condition": "<condition>"
dabei ist <condition> eine Bedingung die erfüllt sein muss, damit der *Value* ermittelt wird. So legt z.B. die <condition> "vendor=[dD]ell*" fest, das der schon erhobene Wert von "vendor" *Dell* oder *dell* enthalten muss.

Hierzu als Beispiel die Klasse "COMPUTER_SYSTEM":

```
{
  "Class": {
    "Type": "STRUCTURAL",
    "Super": [ "HARDWARE_DEVICE" ],
    "Opsi": "COMPUTER_SYSTEM",
    "WMI": "select * from Win32_ComputerSystem",
    "Linux": "[lshw] system"
  },
  "Values": [
    {
      "Type": "varchar(100)",
      "Scope": "i",
      "Opsi": "name",
      "WMI": "Name",
      "Linux": "id"
    },
    {
      "Type": "varchar(50)",
      "Scope": "i",
      "Opsi": "systemType",
      "WMI": "SystemType",
      "Linux": "configuration/chassis"
    },
    {
      "Type": "bigint",
      "Scope": "i",
      "Opsi": "totalPhysicalMemory",
      "WMI": "TotalPhysicalMemory",
      "Linux": "core/memory/size",
      "Unit": "Byte"
    },
    {
      "Type": "varchar(50)",
      "Scope": "i",
      "Opsi": "dellexpresscode",
      "Condition": "vendor=[dD]ell*",
      "Cmd": "#dellexpresscode\dellexpresscode.exe#.split('=')[1]",
      "Python": "str(int('#{COMPUTER_SYSTEM}': 'serialNumber', 'CHASSIS': 'serialNumber')#
    }
  ]
},
```

Besonders interessant ist hier der letzte Value "dellexpresscode":

Dieser ist nur sinnvoll, wenn es sich auch um einen Dell-Rechner handelt, daher die Condition.

Unter Windows wird das Kommandozeilen Programm `dellexpresscode.exe` ausgeführt, welches sich von der `hwaudit.exe` aus gesehen im Unterverzeichnis `dellexpresscode\` befindet. Diese produziert eine Ausgabe in der Form: `dellexpresscode=123456789`. Durch den hinter dem Platzhalter befindlichen `.split('=')[1]` wird der Wert hinter dem Gleichheitszeichen verwendet.

Unter Linux wird geprüft in welchem Element (`COMPUTER_SYSTEM` oder `CHASSIS`) bei `serialNumber` ein Wert gefunden wurde, und dieser dann zur Berechnung des Dell expresscodes verwendet.

Die Opsi-Namen der Values werden über die Dateien `/etc/opsi/hwaudit/locales/*` übersetzt. Bsp. `/etc/opsi/hwaudit/locales/de_DE`:

```
COMPUTER_SYSTEM = Computer
COMPUTER_SYSTEM.systemType = Typ
```

Der Klassenname `COMPUTER_SYSTEM` wird übersetzt in "Computer". Das Opsi-Attribut "systemType" der Klasse `COMPUTER_SYSTEM` wird übersetzt in "Typ". Abschliessend noch der Hinweis: Wenn ein neues Feld erzeugt wird, sollte man dieses in den locale-Dateien anlegen, auch wenn man den Begriff selber nicht übersetzt. Dadurch wird vermieden, dass bei der Laufzeit "Warning" Meldungen produziert werden.

Nachdem Sie die Konfigurationsdatei und die locales modifiziert haben müssen Sie noch den nachfolgenden Aufruftätigen, damit die Änderungen auch in die Datenbank übernommen werden:

```
opsi-setup --init-current-config
```

Weiterhin müssen Sie im `opsi-configd` die Daten komplett neu laden: *Datei/Alle Daten neu laden*.

Der Quellcode diese Paketes ist zu finden auf Github: [opsi-org/hwaudit](https://github.com/opsi-org/hwaudit)

8.3.2 Software Inventarisierung

Die Softwareinventarisierung findet über das Localbootprodukt `swaudit` statt. Dabei werden die Informationen aus dem Uninstallzweig der Registry erhoben und durch zusätzliche Informationen zu Hotfixes und Lizenzkeys ergänzt.

Der Quellcode diese Paketes ist zu finden auf Github: [opsi-org/swaudit](https://github.com/opsi-org/swaudit)

8.4 opsi Abo Produkte

8.4.1 Initiale Bereitstellung von Paketen aus dem Abobereich

Zur Bereitstellung der benötigten Pakete können Sie diese manuell herunterladen oder aber alle Pakete nach Anpassung der `/etc/opsi/opsi-product-updater.conf` mittels `opsi-product-updater -ivv` in einem Rutsch installieren.

Ab opsi Version 4.0.5 können Sie nach Ergänzung der Datei `/etc/opsi/opsi-product-updater.conf` (s.u.)

auch z.B. mittels

```
opsi-product-updater -i -p "mshotfix,mshotfix-win7-x86-glb,mshotfix-win7-win2008r2-x64-glb"
```

die mshotfix-Pakete für Windows 7 erstmalig herunterladen.

8.4.2 Aktualisierung der Abo-Pakete / Konfiguration des opsi-product-updater

Für den weiteren Betrieb ist empfohlen, die Datei `/etc/opsi/opsi-product-updater.conf` zu ergänzen mit den passenden Sektionen, z.B.

```
[repository_abo_mshotfix]
baseUrl = http://download.uib.de
dirs = abo/mshotfix/opsi4/glb
active = false
username = <user>
password = <pass>
autoInstall = false
autoUpdate = true
autoSetup = false
onlyDownload = false

[repository_abo_standard]
baseUrl = http://download.uib.de
dirs = abo/standard/opsi4
```

```

active = false
username = <user>
password = <pass>
autoInstall = false
autoUpdate = true
autoSetup = false
onlyDownload = false

[repository_abo_msoffice]
baseUrl = http://download.uib.de
dirs = abo/msoffice/opsi4
active = false
username = <user>
password = <pass>
autoInstall = false
autoUpdate = true
autoSetup = false
onlyDownload = false

```

und bei den gewünschten Repositories User und Passwort einzutragen sowie die Option `active = true` zu setzen. Für die Aktualisierung der Pakete wird der `opsi-product-updater` empfohlen.



Achtung

Bei Bedarf muss der Proxy in jeder Sektion hinzugefügt werden!

```

; Set Proxy handler like: http://10.10.10.1:8080
proxy =

```

Mittels dem Aufruf `opsi-product-updater -vv` können dann z.B. per Cronjob die installierten Pakete aktualisiert werden.

Eine gleichwertige Variante ist für jedes Repository z.B. eine eigene `/etc/opsi/opsi-product-updater.conf.abo.standard` zu erstellen, die dann mittels `opsi-product-updater -c /etc/opsi/opsi-product-updater.conf.abo.standard` angewendet werden kann.

8.4.3 Festlegung von Default-Properties

Bei der nichtinteraktiven Installation von opsi-Paketen werden auf einem *opsi-configserver* bzw. einem *opsi-server* die Paket default-Properties als default festgelegt.

Falls Sie hiervon abweichende default Properties festlegen wollen so installieren Sie die Pakete einmalig mittels `opsi-package-manager -p ask -i <paketname>.opsi`

Bei einer anschliessenden Installation einer neueren Paketversion mittels `opsi-package-manager` oder `opsi-product-updater` bleiben die eingestellten depot default Properties erhalten.

Ab opsi Version 4.0.5 können die *opsi-server*-default-Properties über das Managementinterface *opsi-configed* festgelegt werden.

8.4.4 Update-Abo MS-Hotfixes

Regelmäßige Updates des Produktes mshotfix (Hotfixes für Windows 7 / 2008-R2, Windows 8.1 / 2012-R2, Windows 10 / Windows 2016).



Achtung

Von Microsoft nicht mehr unterstützte Versionen werden als "failed" angezeigt: 1507 "non"-ItsB, 1511 und 1607 "non"-ItsB ausser Education und Enterprise

Das Update erscheint jeweils innerhalb von 3 Arbeitstagen nach dem Erscheinen eines von Microsoft als wichtig oder kritisch eingeschätzten Patches.

<http://technet.microsoft.com/de-de/security/gg309177.aspx>

<http://technet.microsoft.com/en-us/security/gg309177.aspx>

Bewertung	Beschreibung
Kritisch	<p>Eine Schwachstelle, deren Ausnutzung ohne Eingreifen des Benutzers Codeausführung ermöglichen kann. Dies umfasst \ Szenarien wie sich selbst verbreitende Malware (z. B. Würmer) oder nicht zu vermeidende gebräuchliche \ Anwendungsszenarien, in denen Codeausführung ohne Warnungen oder Eingabeaufforderungen erfolgt. Dies könnte das \ Abrufen einer Webseite oder das Öffnen einer E-Mail bedeuten.</p> <p>Microsoft empfiehlt, dass Kunden kritische Updates unverzüglich installieren.</p> <p>Hoch</p>
Mittel	<p>Eine Schwachstelle, deren Ausnutzung die Vertraulichkeit, Integrität oder Verfügbarkeit von Benutzerdaten oder anderen \ Ressourcen gefährden kann. Dies umfasst gebräuchliche Anwendungsszenarien, in denen der Kunde durch Warnungen oder \ Eingabeaufforderungen unabhängig von deren Herkunft, Qualität oder Nutzbarkeit geschädigt wird. Abfolgen von \ Nutzerhandlungen, die keine Eingabeaufforderungen oder Warnungen hervorrufen, sind ebenfalls abgedeckt.</p> <p>Microsoft empfiehlt, dass Kunden Updates mit hohem Risiko so frühzeitig wie möglich installieren.</p> <p>Mittel</p>
Niedrig	<p>Die Auswirkungen der Schwachstelle sind durch Faktoren wie Authentifizierungsanforderungen oder die Anwendbarkeit nur \ auf Nicht-Standardkonfigurationen erheblich eingeschränkt.</p> <p>Microsoft empfiehlt, dass Kunden die Anwendung des Sicherheitsupdates in Erwägung zieht.</p> <p>Niedrig Die Auswirkungen der Schwachstelle sind aufgrund von Eigenschaften der betroffenen Komponente umfassend \ eingeschränkt. Microsoft empfiehlt, dass Kunden erwägen, ob sie das Sicherheitsupdate auf die betroffenen Systeme \ anwenden.</p>
Rating	Definition
Critical	<p>A vulnerability whose exploitation could allow code execution without user interaction. These scenarios include self-\ propagating malware (e.g. network worms), or unavoidable common use scenarios where code execution occurs without \ warnings or prompts. This could mean browsing to a web page or opening email.</p> <p>Microsoft recommends that customers apply Critical updates immediately.</p> <p>Important</p>
Moderate	<p>A vulnerability whose exploitation could result in compromise of the confidentiality, integrity, or availability of \ user data, or of the integrity or availability of processing resources. These scenarios include common use \ scenarios where client is compromised with warnings or prompts regardless of the prompt's provenance, quality, or \ usability. Sequences of user actions that do not generate prompts or warnings are also covered.</p> <p>Microsoft recommends that customers apply Important updates at the earliest opportunity.</p> <p>Moderate</p>
Low	<p>Impact of the vulnerability is mitigated to a significant degree by factors such as authentication requirements or \ applicability only to non-default configurations.</p> <p>Microsoft recommends that customers consider applying the security update.</p> <p>Low Impact of the vulnerability is comprehensively mitigated by the characteristics of the affected component. \ Microsoft recommends that customers evaluate whether to apply the security update to the affected systems.</p>

Das opsi-mshotfix Paket verwendet (wie WSUS Offline Update <http://forums.wsusoffline.net/-viewtopic.php?f=7&t=172> Abdeckung des / Coverage of WSUS Offline Update) für den Download Microsoft's Update-Katalogdatei wsussen2.cab, um die erforderlichen Patches zu ermitteln. Diese Katalogdatei enthält mindestens alle als „kritisch“ und „sicherheitsrelevant“ eingestuft Updates, aber nicht notwendigerweise alle „wichtigen“ und „optionalen“.

Die opsi mshotfix-Pakete sind modular aufgebaut. Das Basis-Paket „mshotfix“ beinhaltet nur ein Skript zum Installieren der Patches. Die eigentlichen Patches sind gesondert in separaten Paketen enthalten

Tabelle 2: mshotfix Client-Betriebssystem Voraussetzung

OS	
Windows 7 / Windows 2008 R2	Servicepack 1
Windows Windows 2012	Windows 8.1 / Windows 2012 R2
	Windows 10 / Windows 2016

**Achtung**

Wan/VPN Modul Die Pakete für Windows 8.1 / 2012 R2 benötigen opsi-winst \geq opsi-winst_4.11.3.11-1.opsi und opsi-client-agent \geq opsi-client-agent_4.0.4.4-1.opsi

**Achtung**

Wan/VPN Modul Die Pakete für Windows 10 /2016 benötigen opsi-client-agent \geq 4.0.7.9-2

Auf unserem Download-Server ist die Struktur des Abo verzeichnisses

```
mshotfix
!-opsi4/
  !-glb/      Basis-Paket mshotfix sowie globale Patchpakete
              mshotfix-win7-x86-glb,
              mshotfix-win7-win2008r2-x64-glb
              mshotfix-win8-win2012-x64-glb
              mshotfix-win81-x86-glb
              mshotfix-win81-win2012r2-x64-glb
              mshotfix-win10-win2016-x64-glb
              mshotfix-win10-x86-glb
  !-misc/    diverse zusätzliche Pakete
              dotnetfx,
              dotnetfx-hotfix,
              mshotfix-uninstall,
              ms-ie11,
              ms-optional-fixes,
              silverlight
```

Folgende Tabelle soll bei der Auswahl der richtigen Pakete helfen:

Tabelle 3: mshotfix Client-Betriebssystem

OS	Arch. Sprache	Patch-Paket
Windows 7	32Bit	mshotfix-win7-x86-glb
Windows 7	64Bit	mshotfix-win7-win2008r2-x64-glb
Windows 2012	64Bit	mshotfix-win8-win2012-x64-glb
Windows 8.1	32Bit	mshotfix-win81-x86-glb
Windows 8.1	64Bit	mshotfix-win81-win2012r2-x64-glb
Windows 2008 Server R2	64Bit	mshotfix-win7-win2008r2-x64-glb
Windows 2012 R2	64Bit	mshotfix-win81-win2012r2-x64-glb
Windows 10	32Bit	mshotfix-win10-x86-glb
Windows 10	64Bit	mshotfix-win10-win2016-x64-glb
Windows 2016	64Bit	mshotfix-win10-win2016-x64-glb

Installation: `opsi-package-manager -i mshotfix_201008-1.opsi` zum Scharfschalten (überall auf setup stellen, wo das Produkt auf installed steht): `opsi-package-manager -iS mshotfix_201008-1.opsi`

Zusätzlich zum Basis-Paket werden die Patch-Pakete auf die gleiche Weise installiert. Da diese Pakete aber keine Installationskripte enthalten, sind sie nur zusammen mit dem Basis-Paket einzuspielen, d.h. man kann diese nicht separat auf setup stellen. Für die Client-Installation ist komplett das mshotfix-Basispaket zuständig.

Seit Paket `mshotfix 201304-1` werden sich durch `mshotfix installiertePatches` in der Datei `C:\opsi.org\mshotfix\deployed.txt` lokal gemerkt.

Caution

Seit Paket `mshotfix 201808-3` wird zuerst das aktuelle ServingStack installiert mit einem sofortigen Neustart

noreboot

`noreboot=on`: Don't Reboot if possible Warning will be logged if a reboot is required. Will be ignored for Servicing stacks values: ["off", "on"] default: ["off"]

force

`force=on`: All Hotfixes will be installed forced values: ["off", "on"] default: ["off"]

excludes

Commasparated list with kb-numbers or ms-no, that will be excluded (Only Number without beginning kb and no spaces. Example: `123456,789011,2222`) Browser Choice update? (KB976002)

severity

choose the severity that will be installed. Possible Entries are Critical, Important, Moderate, all values: ["Critical", "Important", "Moderate", "all"] default: ["Critical", "Important"]

excludelist-superseded.txt

Use File ExcludeList-superseded.txt values: ["", "ExcludeList-superseded.txt"] default: [""]

monthly-updates

Handle windows-7-and-windows-8-1 : security Only Quality Update vs Monthly Quality Rollup (see [Further simplifying servicing models for Windows 7 and Windows 8.1, More on Windows 7 and Windows 8.1 servicing changes, .NET Framework Monthly Rollups Explained](#)) values: ["all", "monthly_quality_rollup", "security_only_quality_update"] default: ["security_only_quality_update"]

misc mshotfix-uninstall

<code>mshotfix-uninstall</code>	<code>201512-1</code>	MS Hotfix BasePackage
---------------------------------	-----------------------	-----------------------

Entfernt Patches mittels `wusa /uninstall ...` die sich mit dieser Methode deinstallieren lassen.

excludes

Commasparated list with kb-numbers or ms-no, that will be excluded (Only Number without beginning kb and no spaces. Example: `2553154,ms14-082`)

noreboot

`noreboot=on`: Don't Reboot. Warning will be logged if a reboot is required. values: ["off", "on"] default: ["off"]

removefromdeployed.txt

Remove from deployed.txt default: False

removekb

Remove KBXXXXX, (Only Number without beginning kb and no spaces f.e. `3097877`) multivalued: True default: [""]

misc dotnetfx

dotnetfx	4.7.2-3	.NET Framework
----------	---------	----------------

Paket zur Installation der Dotnet Framework Versionen 4.5 und höher insbesondere auf Windows 7 /2008 R2 / 8.1 / 10 / 2012 R2 Auf Windows Versionen (neuer Windows 7) kann auch Dotnet 3.5 installiert werden.

version

["3.5", "4.0", "4.5", "4.5.1", "4.5.2", "4.6", "4.6.1", "4.6.2", "4.7", "4.7.1", "4.7.2", "latest", "latestAnd3.5"] values:
 ["3.5", "4.0", "4.5", "4.5.1", "4.5.2", "4.6", "4.6.1", "4.6.2", "4.7", "4.7.1", "4.7.2", "latest", "latestAnd3.5"] default:
 ["latest"]

rerundotnethotfix::rerun dotnetfx-hotfix after installation if possible values: ["false", "true"] default: ["true"]

install_language_languagepack

install_language_languagepack values: ["auto", "de", "en", "fr"] default: ["auto"]

os-package

Here you can switch from which OS-Version to be install Dotnet3.5, auto=win10 or opsi-local-image-win10 (default); other ProductID for netboot-product values: ["auto",] default: ["auto"]

misc dotnetfx-hotfix

dotnetfx-hotfix	201808-1	dotnetfx-hotfix
-----------------	----------	-----------------

Im Update-Abo *MS-Hotfixes* sind die Hotfixes für *Microsoft .NET Framework* nur enthalten, falls es passend zum jeweiligen Betriebssystem ist.

Also z.B. bei Windows 7 "Microsoft .NET Framework 3.5.1"

(Allerdings gibt es seit Oktober 2016 unter Umständen auch "Monthly Rollups" für DotnetFramework, die im mshotfix-Paket enthalten sind.)

Das Paket dotnetfx-hotfix beinhaltet die Hotfixes von Microsoft für

- Microsoft .NET Framework 4 und höher für Windows 7

Das Paket dotnetfx-hotfix patcht zur Zeit - so vorhanden - die Versionen 4.x auf den jeweils letzten Stand der Reihe.

Bitte beachten Sie: "Support for .NET Framework 4, 4.5, and 4.5.1 ended on January 12, 2016" (<https://support.microsoft.com/en-us/lifecycle?p1=14380>)

bzw. https://support.microsoft.com/en-us/gp/framework_faq/de

noreboot

noreboot=on: Don't Reboot. Warning will be logged if a reboot is required. values: ["off", "on"] default: ["off"]

force

force=on: All Hotfixes will be installed forced values: ["off", "on"] default: ["off"]

severity

choose the severity that will be installed.Possible Entries:Critical, Important, Moderate, all values: ["Critical", "Important", "Moderate", "all"] default: ["all"]

misc ms-ie11

ms-ie11	11.0-11	Internet Explorer 11
---------	---------	----------------------

Win7 Internet Explorer 11

rerunmshotfix

rerun mshotfix after installation values: ["false", "true"] default: ["true"]

client_language

values: ["auto", "de", "en", "fr"] default: ["auto"]

misc ms-optional-fixes

ms-optional-fixes	201808-1	MS optional fixes
-------------------	----------	-------------------

Gedacht als Ergänzung zu mshotfix und basiert auf mshotfix

Enthält:

kb2999226 für win7-win8.1

message_language

Language for messages while installing values: ["auto", "de", "en", "fr"] default: ["auto"]

noreboot

noreboot=on: Don't Reboot if possible Warning will be logged if a reboot is required. Will be ignored for Servicing stacks values: ["off", "on"] default: ["off"]

excludes

Commaseparated list with kb-numbers or ms-no, that will be excluded (Only Number without beginning kb and no spaces. Example: 123456,789011,2222) 49 Browser Choice update? (KB976002) 50 values: [] 51 default: []

rerunmshotfix

rerun mshotfix after installation values: ["false", "true"] default: ["true"]

client_language

values: ["auto", "de", "en", "fr"] default: ["auto"]

misc silverlight

silverlight	5.1.50907.0-1	Microsoft Silverlight
-------------	---------------	-----------------------

8.4.5 Update-Abo MS-Office Hotfixes

Regelmäßige Updates für MS-Office 2010/2013/2016 32 Bit sowie MS-Office 64 Bit

Das Update erscheint jeweils innerhalb von 3 Arbeitstagen nach dem Erscheinen eines von Microsoft als wichtig oder kritisch eingeschätzten Patches.

Tabelle 4: Office hotfix Voraussetzung

Office Version	required
Office 2010 32-bit	Service Pack 2
Office 2013 32-bit	Servicepack 1
Office 2016	

Updates für MS Office 2010 32-bit international: office_2010_hotfix

office_2010_hotfix	201808-1	Microsoft Office 2010 Hotfixes
--------------------	----------	--------------------------------

Enthält sprachunabhängige monatliche Office 2010 Hotfixes (inclusive Visio und Project 2010) . Setzt Servicepack 2 voraus.

Wird getestet gegen Office 2010 Professional und Standard.

Seit Paket office_2010_hotfix 201305-2 werden sich durch office_2010_hotfix installierte Patches in der Datei C:\opsi.org\mshotfix\office_2010_hotfix_deployed.txt lokal gemerkt.

Seit office_2010_hotfix 201503-1:

excludes

Commaseparated list with kb-numbers or ms-no, that will be excluded (Only Number without beginning kb and no spaces. Example: 2553154,ms14-082)

Updates für MS Office 2013 32-bit international: office_2013_hotfix

office_2013_hotfix	201808-1	Microsoft Office 2013 Hotfixes
--------------------	----------	--------------------------------

Enthält sprachunabhängige monatliche Office 2013 Hotfixes (inclusive Visio 2013) . Setzt Servicepack 1 voraus.

Wird getestet gegen Office 2013 Professional

Durch office_2013_hotfix installierte Patches in der Datei C:\opsi.org\mshotfix\office_2013_hotfix_deployed.txt lokal gemerkt.

Seit office_2013_hotfix 201503-1:

excludes

Commaseparated list with kb-numbers or ms-no, that will be excluded (Only Number without beginning kb and no spaces. Example: 2553154,ms14-082)

Updates für MS Office 2016 32-bit international: office_2016_hotfix

office_2016_hotfix	201808-1	Microsoft Office 2016 Hotfixes
--------------------	----------	--------------------------------

Enthält sprachunabhängige monatliche Office 2016 Hotfixes (inclusive Visio 2016) .

Wird getestet gegen Office 2016 Professional

Durch office_2016_hotfix installierte Patches in der Datei

C:\opsi.org\mshotfix\office_2016_hotfix_deployed.txt

lokal gemerkt.

excludes

Commaseparated list with kb-numbers or ms-no, that will be excluded (Only Number without beginning kb and no spaces. Example: 2553154,ms14-082)

CAUTION

Falls Sie Updates für MS Office 2016 32-bit und 64-Bit oder aber nur 64-Bit einsetzen wollen:

Passen Sie Ihre opsi-oproduct-updater.conf an: x3264 / x64

8.4.6 Update-Abo *opsi Standardprodukte*

Regelmäßige Updates der Produkte:

```
Adobe Reader DC Classic / Continuous (international / 32 Bit)
Adobe Flashplayer (international / 32 Bit / 64 Bit)
Apache OpenOffice.org (deutsch / 32 Bit)
Google Chromium for business ( international 32 Bit / 64 Bit )
LibreOffice ( international 32 Bit / 64 Bit )
Mozilla Firefox (deutsch, englisch, französisch und niederländisch. / 32 Bit) bzw. 32/
Mozilla Thunderbird (deutsch, englisch und französisch / 32 Bit)
Oracle Java VM (international / 32 Bit / 64 Bit)
```

Je nach Vertrag stellen wir noch folgende Sprachen im Abo zur Verfügung:

```
Mozilla Firefox (zusätzlich dänisch, italienisch, spanisch, tschechisch und norwegisch)
Mozilla Thunderbird (zusätzlich italienisch / 32 Bit)
```

weitere Sprachen auf Anfrage.

Das Update erscheint jeweils innerhalb von 2 Arbeitswochen nach dem Erscheinen eines Updates dieser Produkte; bei vom Hersteller als kritisch eingestuften Security-Updates innerhalb von 1 Arbeitswoche.

Customizing der Pakete durch zentrale Konfigurationen

Für die Pakete

```
adobe.reader.dc.classic
adobe.reader.dc.continuous
firefox
flashplayer
javavm
thunderbird
```

gibt es die Möglichkeit eigene Konfigurationen zu erstellen und im Verzeichnis `custom` zu hinterlegen, die über Properties auswählbar sind. (Näheres siehe unten)

Customizing der Pakete durch preinst/postinst-scripts

Für die Pakete

```
adobe.reader.dc.classic
firefox          (ab 17.0.6 esrorstandard -1)
flashplayer      (ab 13.0.0.182 or 11.7.700.275 -1)
google-chrome-for-business
javavm           (ab javavm_1.7.0.51 -5 )
libreoffice      (ab libreoffice_4.3.5 or 4.4.0 -2)
ooffice         (ab ooffice_4.1.1 -2)
thunderbird      (ab 17.0.6 esrorstandard -1)
```

besteht die Möglichkeit eigene custom-Scripte einzubauen im Verzeichnis `custom\scripts`

Einfache Templates für die unterstützen Scripte finden sich im Verzeichnis `opsi\scripts`

```

custom.actions.post.setup
custom.actions.post.uninstall
custom.actions.pre.setup
custom.actions.pre.uninstall
custom.declarations
custom.sections

custom scripts will be included at
- setup-script
- uninstall-script

custom pre-scripts will be included in
- setup-script
- uninstall-script

custom post-scripts will be included in
- in setup-script
- uninstall-script

custom.declarations
; intended for declaration of custom Variables and Stringlist Variables
; will be included with "include_insert" at top of [actions]
; but after GetProductProperties

custom.sections
; intended for declaration of custom secondary sections
; will be included with "include_append" at top of [actions]
; but after GetProductProperties

custom.actions.pre.setup (or custom.actions.pre.uninstall)
; will be included with "include_insert" at at top of [actions]
; (but after GetProductProperties)

custom.actions.post.setup (or custom.actions.post.uninstall)
; will be included with "include_insert" in case of successful installation before "endof_"actions"
; in setup-script ( or uninstall-script)

```

Adobe Acrobat Document Cloud Classic : adobe.reader.dc.classic

adobe.reader.dc.classic

20151500630448 or 20171701130099 -1 Adobe

Das adobe.reader.dc.classic-Paket beinhaltet Adobe Acrobat Document Cloud Classic (MUI-Version)

Adaptation in the transform file *.mst

```

cat transform.txt
Changes vs default the transform file *.mst

Personalisation Options
Suppress Eula

Installation Options
activated - Make Reader the default PDF viewer
IF REBOOT REQUIRED - suppress reboot

Shortcuts
deactivated - Destination Computer/Desktop/Adobe Reader XI (Icon)

Online and Acrobat.com Features
Online Features
activated - Disable product updates
Enable & Ask before Installing - Load trustet root certificates from Adobe

```

```

Online Services and Features
disable product updates
Load trusted root certificates from Adobe disable
DISABLE all Services

```

adobereader.mst

Das Adobe Reader Paket von uib verwendet eine Standard-transform-Datei die mit dem Adobe Customization Wizard erstellt ist. Abweichend davon können im Share `opsi_depot` im Verzeichnis `/var/lib/opsi/depot/adobe.reader.dc.classic/custom` eigene MST-Dateien abgelegt werden, die über diese Property ausgewählt werden können. Beim Einspielen des `adobe.reader.dc.classic`-Paketes auf dem opsi-Server wird das Verzeichnis `custom` ueber ein `preinst/postinst`-Script gesichert.



Achtung

Bei Einsatz des Wan/VPN Moduls muss nach Änderungen im Verzeichnis `custom` das Paket neu eingespielt werden, damit die Datei `<productid>.files` neu erzeugt wird.

client_language

Das `adobe.reader.dc.classic`-Paket als MUI Ist der Wert auf "auto" gesetzt, so wird anhand der Systemsprache automatisch die Sprache erkannt. values: ["auto", "de", "en", "fr"] default: ["auto"]

classicversion

description: <https://helpx.adobe.com/acrobat/release-note/release-notes-acrobat-reader.html> Classic Track (2015 Release) or (2017 Release) values: ["2015", "2017"] default: ["2015"]

noreboot

description: `noreboot=true`: Don't Reboot. Warning will be logged if a reboot is required. values: ["false", "true"] default: ["false"]

Adobe Acrobat Document Cloud Continuous : `adobe.reader.dc.continuous`

<code>adobe.reader.dc.continuous</code>	201801120058-1
Adobe acrobat reader dc continuos	

Das `adobe.reader.dc.continuous`-Paket beinhaltet Adobe Acrobat Document Cloud Continuous (MUI-Version)

Adaptation in the transform file *.mst

```

cat transform.txt
Changes vs default the transform file *.mst

Personalisation Options
Suppress Eula

Installation Options
activated - Make Reader the default PDF viewer
IF REBOOT REQUIRED - suppress reboot

Shortcuts
deactivated - Destination Computer/Desktop/Adobe Reader (Icon)

Online and Acrobat.com Features
Online Features
activated - Disable product updates
Enable & Ask before Installing - Load trustet root certificates from Adobe

```

```
Online Services and Features
disable product updates
Load trusted root certificates from Adobe disable
DISABLE all Services
```

adobereader.mst

Das Adobe Reader Paket von uib verwendet eine Standard-transform-Datei die mit dem Adobe Customization Wizard erstellt ist. Abweichend davon können im Share `opsi_depot` im Verzeichnis `/var/lib/opsi/depot/adobe.reader.dc.continuous/custom` eigene MST-Dateien abgelegt werden, die über diese Property ausgewählt werden können. Beim Einspielen des `adobe.reader.dc.continuous`-Paketes auf dem opsi-Server wird das Verzeichnis `custom` ueber ein `preinst/postinst`-Script gesichert.



Achtung

Bei Einsatz des Wan/VPN Moduls muss nach Änderungen im Verzeichnis `custom` das Paket neu eingespielt werden, damit die Datei `<productid>.files` neu erzeugt wird.

client_language

Das `adobe.reader.dc.continuous`-Paket als MUI Ist der Wert auf "auto" gesetzt, so wird anhand der Systemsprache automatisch die Sprache erkannt. values: ["auto", "de", "en", "fr"] default: ["auto"]

noreboot

description: `noreboot=true`: Don't Reboot. Warning will be logged if a reboot is required. values: ["false", "true"] default: ["false"]

Adobe Flashplayer : flashplayer

flashplayer	30.0.0.154-1 Adobe Flashplayer
-------------	--------------------------------

Das `flashplayer`-Paket beinhaltet Adobe Flashplayer in den von Adobe unterstützten Versionen `standard` und `esr` (Extended Support Release <http://blogs.adobe.com/flashplayer/2014/03/upcoming-changes-to-flash-players-extended-support-release.html>)

flashplayer-version

Flashplayer `standard` as default or Extended Support Release (`esr`) <http://blogs.adobe.com/flashplayer/2014/03/upcoming-changes-to-flash-players-extended-support-release.html> values: `esr`, `standard` default: `standard`

Die zentrale Konfigurationsdatei `mms.cfg`

wird bei jedem neu "auf setup" setzen

neu erstellt und gemaess den Produktproperties angepasst (Ausnahme: Verwendung einer `custom mms.cfg`)

Bei leeren Werten werden keine Eintraege gemacht.

Seit Version `13.0.0.250-2` besteht die Möglichkeit abweichend davon eine `custom mms.cfg` bereitzustellen

mms.cfg

`/var/lib/opsi/depot/flashplayer/custom` eigene `mms.cfg` -Dateien abzulegen, die über diese Property ausgewählt werden können.

Beim Einspielen des `flashplayer`-Paketes auf dem opsi-Server wird das Verzeichnis `custom` ueber ein `preinst/postinst`-Script gesichert.

+ CAUTION: Bei Einsatz des Wan/VPN Moduls muss nach Änderungen im Verzeichnis `custom` das Paket neu eingespielt werden, damit die Datei `<productid>.files` neu erzeugt wird.

Weitere Einstellungen ausser Verwendung der Property autoupdatedisable::

werden dann nicht verwendet

Weitere Hinweise siehe

Adobe Flashplayer Admin Guide:

flash_player_11_1_admin_guide.pdf (flash_player_admin_guide.pdf)

#####

Chapter 4: Administration

You can create and place files on the end user's machine to manage features related to security, privacy, use of disk space, and so on.

Privacy and security settings (mms.cfg)

As a network administrator, you can install Flash Player across the enterprise while enforcing some common global security and privacy settings (supported with installation-time configuration choices). To do this, you install a file named mms.cfg on each client machine.

The mms.cfg file is a text file. When Flash Player starts, it reads its settings from this file, and uses them to manage functionality as described in the following sections.

mms.cfg file location

Assuming a default Windows installation, Flash Player looks for the mms.cfg file in the following system directories:

32-bit Windows - %WINDIR%\System32\Macromed\Flash

64-bit Windows - %WINDIR%\SysWow64\Macromed\Flash

Note: The %WINDIR% location represents the Windows system directory, such as C:\WINDOWS.

assetcachesize

description: hard limit, in MB, on the amount of local storage that Flash Player uses for the storage of common Flash components values: ["20"] default: ["20"]

autoupdatedisable

description: Lets you prevent Flash Player from automatically checking for and installing updated versions. values: ["0", "1"] default: ["1"]

autoupdateinterval

description: (without meaning if AutoUpdateDisable=1) how often to check for an updated version of Flash Player

avhardwareisable

description: Lets you prevent SWF files from accessing webcams or microphones values: ["0", "1"] default: ["1"]

disabledevicefontenumeration

description: Lets you prevent information on installed fonts from being displayed. values: ["0", "1"] default: ["1"]

fullscreenisable

description: Lets you disable SWF files playing via a browser plug-in from being displayed in full-screen mode values: ["0", "1"] default: ["1"]

localfilereaddisable

description: Lets you prevent local SWF files from having read access to files on local hard drives values: ["0", "1"] default: ["0"]

filedownloaddisable

description: Lets you prevent the ActionScript FileReference API from performing file downloads values: ["0", "1"] default: ["0"]

fileuploaddisable

description: Lets you prevent the ActionScript FileReference API from performing file uploads values: ["0", "1"] default: ["0"]

disableproductdownload

description: Lets you prevent native code applications that are digitally signed and delivered by Adobe from being downloaded values: ["0", "1"] default: ["1"]

disablesockets

description: enable or disable the use of the Socket.connect() and XMLSocket.connect() methods values: ["0", "1"] default: ["1"]

enablesocketsto

description: Lets you create a whitelist of servers to which socket connections are allowed

enforcelocalecurityinactivexhostapp

description: Lets you enforce local security rules for a specified application.

legacydomainmatching

description: Lets you specify whether SWF files produced for Flash Player 6 and earlier can execute an operation that has been restricted in a newer version of Flash Player values: ["0", "1"] default: ["0"]

localfilelegacyaction

description: Lets you specify how Flash Player determines whether to execute certain local SWF files that were originally produced for Flash Player 7 and earlier values: ["0", "1"] default: ["0"]

allowuserlocaltrust

description: Lets you prevent users from designating any files on local file systems values: ["0", "1"] default: ["0"]

localstoragelimit

description: Lets you specify a hard limit on the amount of local storage that Flash Player uses (per domain) for persistent shared objects. values: ["3"] default: ["3"]

overridegpuvalidation

description: Overrides validation of the requirements needed to implement GPU compositing values: ["0", "1"] default: ["0"]

rtmfpp2pdisable

description: Specifies how the NetStream constructor connects to a server when a value is specified for peerID, the second parameter passed to the constructor values: ["0", "1"] default: ["1"]

disablenetworkandfilesysteminhostapp

description: 1 (Acrobat.exe,Acroread.exe,WINWORD.EXE,EXCEL.EXE,POWERPNT.EXE,PPTVIEW.EXE,OUTL... values: ["0", "1"] default: ["1"]

- **Bekannte Probleme**

- Beim Einspielen "On Demand" kann die Installation aufgrund geöffneter Browser fehlschlagen!

Google Chromium for Business

```
google-chrome-for-business 56.0.2924.76-1
```

Das Paket beinhaltet den msi-Installer von Google (Häufig gestellte Fragen Chrome for Business <https://support.google.com/a/answer/188447>).

Bemerkungen:

Die Deinstallation und Installation des google-chrome.msi schlägt manchmal fehl.

Daher gibt es verschiedene Ansätze im opsi-Paket, um die Zuverlässigkeit der Installation zu erhöhen.

Ein Kunde berichtete von einer Erfolgsquote von 100% bei 40 Installationen mit folgender Einstellung der Properties:

- `install_architecture`: 32
- `reboot_on_retry`: True
- `reboot_after_uninstall`: True
- `timeout`: 240

Intern verwenden wir für Tests: `* install_architecture: system specific * reboot_on_retry: True * reboot_after_uninstall: True * timeout: notimeout`

autoupdate

!Property wirkt nicht mehr!

vgl. <https://support.google.com/chrome/a/answer/187207>

gibt es 0,1,2,3 als Wert für HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Update\UpdateDefault bzw ADM falls Gruppenrichtlinien verwendet werden

ADM= use Policy based on Googles Template, 0=UpdatesDisabled, 1=UpdatesEnabled, 2=ManualUpdatesOnly, 3=AutomaticUpdatesOnly, values: ["0", "1", "2", "3", "ADM"] default: ["0"]

removeupdatehelper

default: ["true"]

install_architecture

description: which architecture (32/64 bit) has to be installed values: ["32", "64", "system specific"] default: ["system specific"]

reboot_on_retry

description: If installation fails and (timeout > 0) then reboot before retry default: False

reboot_after_uninstall

description: reboot after uninstall old version default: False

timeout

description: TimeoutSeconds msi installs values: ["240", "300", "600", "notimeout"] default: ["notimeout"]

Apache OpenOffice : ooffice4

ooffice	4.1.1 – 5	Apache OpenOffice
---------	-----------	-------------------

Das ooffice-Paket beinhaltet Apache OpenOffice in deutsch.

handle_excel_xls

Soll OpenOffice.org fuer MS-Excel-Dokumente registriert werden on = OpenOffice.org wird fuer MS-Exceldokumente registriert off = OpenOffice.org wird nicht fuer MS-Exceldokumente registriert

handle_powerpoint_ppt

Soll OpenOffice.org fuer MS-PowerPoint-Dokumente registriert werden on = OpenOffice.org wird fuer MS-PowerPointdokumente registriert off = OpenOffice.org wird nicht fuer MS-PowerPointdokumente registriert

handle_word_doc

Soll OpenOffice.org fuer MS-Word-Dokumente registriert werden on = OpenOffice.org wird fuer MS-Worddokumente registriert off = OpenOffice.org wird nicht fuer MS-Worddokumente registriert

LibreOffice The Document Foundation : libreoffice

libreoffice	6.0.6 or 6.1.0-2	libreoffice
-------------	------------------	-------------

Das libreoffice-Paket beinhaltet LibreOffice 6 international.

client_language

client_language - only for messages important, cause libre office is international values: ["auto", "de", "en", "fr"]

msoregister

Open Microsoft Office documents with LibreOffice (true) values: ["false", "true"] default: ["false"]

libreoffice-version

description: *Stable* - is an Extended Support Release from LibreOffice for the conservative user - default version (6.0.6) *Experimentell* is a version for the experimentell user from LibreOffice (6.1.0) values: ["experimentell", "stable"] default: ["stable"]

hide_component

description: Hide component base by removing desctoplinc and exe file values: ["base", "none"] default: ["none"]

ui_languages

description: which UI languages should be installed (comma separated), For example UI_LANGS=en_US,de,fr,hu will install English (US), German, French, and Hungarian. default: ["auto"]

install_architecture

which architecture (32/64 bit) has to be installed values: ["32", "64", "system specific"] default: ["32"]

Mozilla Firefox : firefox

firefox	60.2.2 esr or 62.0.3-2
---------	------------------------

Das firefox-Paket beinhaltet Mozilla Firefox in den Sprachen deutsch, englisch, französisch und niederländisch.

Es werden alle von Mozilla unterstützten Versionen bereitgestellt.

Firefox kann zentral konfiguriert werden a) entweder über eine zentrale Konfigurationsdatei `mozilla.cfg` (vgl. http://kb.mozillazine.org/Locking_preferences) b) oder über eine `policies.json` (vgl. <https://github.com/mozilla/policy-templates/blob/master/README.md>)

die in folgendem Verzeichnis abzulegen ist:

```
/var/lib/opsi/depot/firefox/custom/
```

Bei erneutem Einspielen des Pakets werden die gefundenen Konfigurationsdateien über preinst-/postinst-Mechanismus erhalten und dann über das Property "mozillacfg" auswählbar.

- Beispiel:

```
cat /var/lib/opsi/depot/firefox/custom/mozilla.cfg
//
lockPref("browser.startup.homepage", "http://www.uib.de");
lockPref("network.proxy.type", 1);
lockPref("network.proxy.http", "router.uib.local");
lockPref("network.proxy.http_port", 3128);
```

Alternativ zu einer mozilla.cfg kann man auch eine mit dem CCK2 erstellte autoconfig.zip über das Property "mozil-lacfg" einbinden.

**Achtung**

Bei Einsatz des Wan/VPN Moduls muss nach Änderungen im Verzeichnis custom das Paket neu eingespielt werden, damit die Datei <productid>.files neu erzeugt wird.

client_language

values: ["auto", "de", "en", "fr", "nl"] default: ["auto"]

firefox-version

Firefox *esr* - is the Extended Support Release from Mozilla.org (60.2.2esr); (62.0.3) is the Mozilla.org standard version

values: ["esr", "standard"] default: ["esr"]

pref_file

(user/prefs)= in welcher Datei sollen die Einstellungen gemacht werden user.js oder prefs.js. values: ["prefs", "user"] default: ["prefs"]

noautoupdate

(on/off): auto update ausschalten. default=on

setproxy

(off/direct/manual/file) proxy einstellungen verändern

- off= tue nichts
- direct = direkte Verbindung zum Internet
- manual = Proxyeinstellungen über property proxysetting (<ip-numme>:<port>) und property no-proxy_hosts (host1,host2)
- file = Proxyeinstellungen über property proxysetting (<path_to_proxyconf.pac>) und property no-proxy_hosts (host1,host2)
- system
- default=off

proxysetting

string für proxy Einstellung (siehe: setproxy)

noproxy_hosts

Komma separierte Liste von hosts

mozillacfg

description: filename for mozilla.cfg in %scriptpath%\custom-directory, http://kb.mozillazine.org/Locking_preferences

profilemigrator

enable or disable Profilemigrator on first run values: ["off", "on"] default: ["off"]

- **Bekannte Probleme**

- Beim Einspielen "On Demand" kann die Installation aufgrund geöffneter Programme fehlschlagen!

Mozilla Thunderbird : thunderbird

thunderbird	60.2.1 – 1
-------------	------------

Das thunderbird-Paket beinhaltet Mozilla Thunderbird in den Sprachen deutsch, englisch und französisch.

Es werden alle von Mozilla unterstützten Versionen bereitgestellt.

Analog dem Firefox-Paket kann zentrale Konfigurationsdatei bereitgestellt werden.

client_language

values: ["auto", "de", "en", "fr"] default: ["auto"]

thunderbird-version

values: ["45.x"] default: ["45.x"]

addonsactivation

description: Enable/Disable AddOns (default = enable) values: ["off", "on"] default: ["on"]

https://developer.mozilla.org/en/Addons/Add-on_Manager/AddOnManager

<http://mike.kaply.com/2012/02/09/integrating-add-ons-into-firefox/>

https://developer.mozilla.org/en/Thunderbird/Deploying_Thunderbird_in_the_Enterprise/Thunderbird_Preferences_Relevant_to_Enterprises

```
Set_Netscape_User_Pref ("extensions.autoDisableScopes", 11)
Set_Netscape_User_Pref ("extensions.shownSelectionUI", true)
```

enigmail

description: Install GnuPG-Plugin values: ["off", "on"] default: ["off"]

noautoupdate

description: disable automatic updates values: ["off", "on"] default: ["on"]

mozillacfg

description: filename for mozilla.cfg in %scriptpath%\custom-directory, http://kb.mozillazine.org/Locking_preferences

lightning

description: Install calendar plugin lightning values: ["off", "on"] default: ["off"]

- **Bekannte Probleme**

– Beim Einspielen "On Demand" kann die Installation aufgrund geöffneter Programme fehlschlagen!

Oracle Jre : javavm

javavm	1.8.0.181 – 1	Oracle Java Run
--------	---------------	-----------------

Das javavm-Paket beinhaltet die Oracle Jre 8.x (cpu und psu, falls verfügbar) Für die Unterscheidung zwischen "cpu" und "psu" siehe <http://www.oracle.com/technetwork/java/javase/downloads/cpu-psu-explained-2331472.html>

(Für Jre 1.6.x ist das Ende der "Public Updates" für in 2013 angekündigt, <http://www.oracle.com/technetwork/java/eol-135779.html>)

Seit JRE 1.8u20 scheint JRE 8 nur noch im static Mode installierbar.

<http://www.oracle.com/technetwork/java/javase/jre-install-137694.html>

Vor der Installation werden bestehen "Patch-in-place" Jre-Versionen 1.6.x, 7.x und 1.8x deinstalliert, seit 1.8u20 auch die static-Versionen.

- Vorhandene Oracle JRE mit "Static configuration" werden nicht deinstalliert.
- Eventuell vorhandene Oracle JRE (version 1.6.0 - version 1.6.7) werden deinstalliert, es sei denn Sie werden explizit über die property "keepversion" ausgeschlossen oder aber die Property `uninstalljava16=false` gesetzt.

Eine zentrale Konfigurationsdatei `deployment.properties` (vgl. <http://docs.oracle.com/javase/6/docs/technotes/-guides/deployment/deployment-guide/properties.html> <http://docs.oracle.com/javase/7/docs/technotes/guides/-deployment/deployment-guide/properties.html>) kann bereitgestellt werden im Verzeichnis

```
/var/lib/opsi/depot/javavm/custom/
```

Bei erneutem Einspielen des Pakets werden die gefundenen Konfigurationsdateien über `preinst-/postinst-`Mechanismus erhalten und dann über die Property "deployment.properties" auswählbar.

- Beispiel:

```
cat /var/lib/opsi/depot/javavm/custom/deployment.properties
//
#deployment.properties
#Tue Dec 18 12:36:01 CET 2012
deployment.javaws.autodownload=NEVER
deployment.javaws.autodownload.locked=
deployment.security.validation.ocsp=true
deployment.security.validation.ocsp.locked=
deployment.security.validation.crl=true
deployment.security.validation.crl.locked=
```



Achtung

Bei Einsatz des Wan/VPN Moduls muss nach Änderungen im Verzeichnis `custom` das Paket neu eingespielt werden, damit die Datei `<productid>.files` neu erzeugt wird.

install_architecture

description: which architecture (32/64 bit) has to be installed values: ["32 only", "64 only", "both", "system specific"] default: ["both"]

javaversion

description: which version has to be installed (JRE 8 = 1.8.x); jre8 1.8.0_121-b13 (CPU); jre8psu 1.8.0_121-b13 (CPU)

values: ["jre8", "jre8psu"] default: ["jre8"]

keepversion

description: Dont uninstall jre version values: ["1.6.0_0", "1.6.0_1", "1.6.0_2", "1.6.0_3", "1.6.0_4", "1.6.0_5", "1.6.0_6", "1.6.0_7", "none"] default: ["none"]

webjava

description: TESTING: http://java.com/en/download/help/silent_install.xml WEB_JAVA=0, if used, disables any Java application from running in the browser values: ["0", "1", "none"] default: ["none"]



Achtung

WEB_JAVA=0 liefert missverständliche Meldungen im Browser ("fehlendes plugin")

uninstalljava16

description: Uninstall Java 1.6 "Patch in Place" Installations default: True

uninstalljava17

description: Uninstall Java 1.7 "Patch in Place" Installations default: True

deployment.properties

description: filename for deployment.properties in %scriptpath%\custom-directory, <http://docs.oracle.com/javase/7/docs/technotes/guides/deployment/deployment-guide/properties.html> saved by pre- postinstscript

startmenuentry

Create Entry for java in common startmenu aboutJava,CheckForUpdates,ConfigureJava,GetHelp,VisitJava.com
default: False

timeout

TimeoutSeconds msi installs values: ["180", "240", "300", "notimeout"] default: ["300"]

copy.java.exe.to.system32

Copy java.exe,javaw.exe, javaws.exe to system32 default: False

environment_set_java_home

Set the environment variable JAVA_HOME default: False

- **Bekannte Probleme**

- Beim Einspielen "On Demand" kann die Installation aufgrund geöffneter Programme fehlschlagen!

9 opsi Erweiterungen

9.1 Freischaltung kostenpflichtiger Module

Auch wenn opsi Open Source ist, so gibt es einige Zusatzkomponenten, die im Rahmen eines Kofinanzierungsprojektes erstellt wurden bzw. gepflegt werden und (noch) nicht kostenlos sind.

Zur Zeit (Dezember 2016) sind dies:

- MySQL-Backend für Konfigurationsdaten (siehe Abschnitt [5.6.2](#))
- UEFI Support (siehe Abschnitt [9.6](#))
- opsi-Lizenzmanagement Modul (siehe Abschnitt [9.9](#))
- Unterstützung für Clients über WAN/VPN (siehe Abschnitt [9.10](#))
- opsi WIM Capture (siehe Abschnitt [9.4](#))
- opsi Local Image (siehe Abschnitt [9.7](#)) und opsi-vhd-reset (siehe Abschnitt [9.8](#))
- opsi Linux Agent (siehe Abschnitt [9.5](#))
- opsi Nagios Connector (siehe Abschnitt [9.11](#))
- User-Rollen (siehe Abschnitt [4.17.1](#))
- die Scalability1-Erweiterung zur Performance-Erhöhung bei sehr großen Installationen

Zu diesem Thema siehe auch: <http://uib.de/www/kofinanziert/index.html>

Solange die Zusatzkomponenten den Cofunding-Status besitzen, können Sie nur zur Evaluation frei genutzt werden, zur dauerhaften regulären Verwendung sind jedoch die Cofunding-Beiträge zu zahlen.

Die Zulässigkeit der Verwendung von Modulen ist auf dem opsi-Server in der Freischaltdatei `/etc/opsi/modules` festgelegt. Es handelt sich um eine einfache Textdatei mit der Information, welches Modul (für wie viele Clients) aktiviert ist. Die Datei ist gegen Veränderung geschützt durch eine digitale Signatur. Bei fehlenden Angaben wird von den Defaultwerten ausgegangen. Fehlt die Freischaltdatei komplett, sind nur die standardmäßigen freien Komponenten von opsi aktiv. Bei einer befristeten Freischaltung ist in der Datei die Befristung enthalten.

Um zu Evaluierungszwecken eine zeitlich befristet gültige Freischaltdatei zu erhalten, wenden Sie sich an info@uib.de. Im Rahmen einer Beteiligung an den entsprechenden Kofinanzierungsprojekten erhalten Sie eine Freischaltdatei zur dauerhaften und regulären Nutzung der freigeschalteten Komponenten.

Wenn Sie eine `modules`-Datei erhalten haben, kopieren Sie sie nach `/etc/opsi`

Führen Sie danach den folgenden Befehl aus:

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

Starten Sie danach noch den `opsiconfd` neu.

Kontrollieren Sie die Freischaltung mit einer der folgenden Methoden:

Im opsi-configed können Sie über den Menüpunkt Hilfe/opsi-Module sich den Status Ihrer Freischaltung anzeigen lassen.

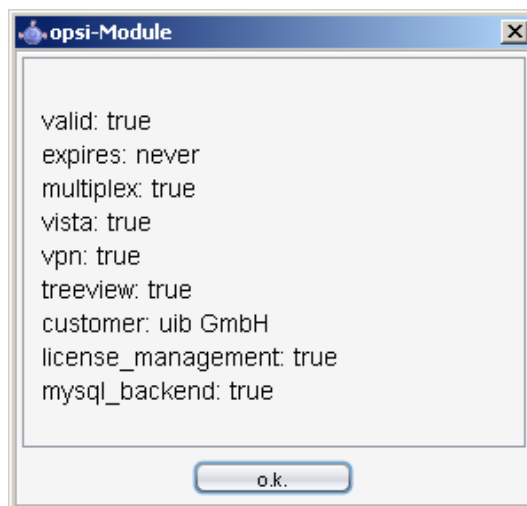


Abbildung 79: Anzeige der Freischaltung im opsi-configed

Mit der Methode `backend_info` können Sie mit `opsi-admin` überprüfen, welche Module freigeschaltet sind. (Hinweis: Geben Sie weder die Datei noch die Ausgabe dieses Befehls öffentlich weiter, zumindest nicht ohne die Signatur zu löschen).

```
opsi-admin -d method backend_info
{
  "opsiVersion" : "3.99.0.0",
  "modules" :
  {
    "customer" : "uib GmbH",
    "vista" : true,
    "vpn" : true,
    "license_management" : true,
    "expires" : "never",
    "valid" : true,
```

```
"multiplex" : true,  
"signature" : "DIES-IST-KEINE-GÜLTIGE-SIGNATUR",  
"treeview" : true,  
"mysql_backend" : true  
}  
}
```

9.2 User-Rollen (via opsi-configed)

Es muss das Feature *user roles* in der *modules*-Datei freigeschaltet sein. Zur Bedienung siehe Abschnitt [4.17.1](#)

9.3 opsi directory connector

9.3.1 Einführung

Der opsi Directory Connector ist ein Werkzeug um Daten aus einem Verzeichnisdienst in eine opsi-Installation zu überführen. Dadurch wird mehrfacher Pflegeaufwand in unterschiedlichen Systemen vermieden.

9.3.2 Vorbedingungen für die opsi Erweiterung *opsi directory connector*

Dieses Modul ist momentan eine [kofinanzierte opsi Erweiterung](#).

Es sind eine Reihe von Vorbedingungen nötig, um dieses Modul einsetzen zu können. Das bedeutet, dass Sie zum Einsatz eine Freischaltdatei benötigen. Diese Freischaltung erhalten Sie, wenn Sie die Erweiterung kaufen. Zu Evaluierungszwecken stellen wir Ihnen auch eine zeitlich befristete Freischaltung kostenlos zur Verfügung (→ mail an info@uib.de).

Allgemeine Anforderungen

Der Quell-Verzeichnisdienst muss das LDAP-Protokoll implementieren.

Das Ziel-Opsi-System sollte mindestens opsi 4.0.7 verwenden. Ältere Versionen können funktionieren, wurden aber nicht getestet.

Die Maschine, auf welcher der Connector laufen soll, muss über das Netzwerk Zugriff auf den Directory- und opsi-Server haben. Es ist möglich alle Komponenten auf der gleichen Maschine zu betreiben, aber es wird davon ausgegangen, dass jeweils getrennte Maschinen verwendet werden.

Hardware-Anforderungen

Diese Anforderungen richten sich an eine einfache Verwendung in einer kleinen Umgebung mit bis zu 500 Clients. Diese Anforderungen fallen in großen Umgebungen gegebenenfalls größer aus, weshalb Anpassungen notwendig sein können.

- 256 MB freier Arbeitsspeicher
- Netzwerkverbindungen

Software-Anforderungen

Es wird nur die Installation und der Betrieb des Connectors unter Linux unterstützt. Eine Unterstützung für Windows ist nicht geplant.

Der Connector verwendet Python 3, welches mindestens in Version 3.2 vorliegen muss.

Durch die Verwendung standardisierter Protokolle zur Kommunikation werden keine zusätzlichen opsi- oder Verzeichnisdienst-spezifischen Komponenten benötigt.

9.3.3 Installation

Zur Installation fügen Sie bitte das opsi-Repository wie im Dokument *Getting Started* beschrieben hinzu.

Anschließend verwenden Sie den Paket-Manager des Betriebssystems um das Paket `opsi-directory-connector` zu installieren.

Auf einer Debian-basierten Maschine kann die Installation wie folgt durchgeführt werden:

```
apt-get install opsi-directory-connector
```

Anmerkung

CentOS und RedHat stellen in Version 6 und 7 kein Python 3 als Teil ihrer Kern-Repositories bereit, weshalb die Installation auf diesen Maschinen von uns nicht unterstützt wird.

9.3.4 Konfiguration

Der Connector kann über eine Vielzahl an Einstellungsmöglichkeiten an verschiedenste Umgebungen angepasst werden.

Die Konfiguration geschieht über eine Konfigurationsdatei im JSON-Format, welche gültiges JSON enthalten muss. Zur Angabe von booleschen Werten verwenden Sie bitte `true` oder `false`. Text muss mit doppelten Anführungszeichen eingegeben werden, beispielsweise `"das ist Text"`.

Eine Beispiel-Konfiguration wird unter `/etc/opsi/opsidirectoryconnector.example.conf` bereitgestellt. Diese Datei kann als eine Vorlage für eigene Konfigurationen verwendet werden.

```
cp /etc/opsi/opsidirectoryconnector.example.conf /etc/opsi/opsidirectoryconnector-custom.conf
```

Directory-Einstellungen

Diese Einstellungen werden benötigt, um eine Verbindung zum Verzeichnisdienst herzustellen und den Suchbereich auf bestimmte Bereiche und Objekte einzugrenzen.

```
{
  "directory": {
    "address": "ldap://192.168.12.34",
    "user": "DOMAIN\\opsiconnector",
    "password": "insertpasswordhere",
    "passwordFile": "",
    "search_base": "dc=testcompy,dc=local",
    "search_query_computers": "(objectClass=computer)",
    "search_query_groups": "(objectClass=organizationalUnit)",
    "connection_options": {
      "start_tls": true,
      "paged_search_limit": 768
    }
  },
  ...
}
```

Unter `address` muss angegeben werden unter welcher Adresse der Server angesprochen wird. `user` und `password` werden für die Authentifikation an Selbigem verwendet. Sofern für `passwordFile` ein Wert angegeben wird, wird dieser als Pfad zu einer Datei, welche das Passwort enthält, interpretiert. Der Inhalt dieser Datei wird als Passwort verwendet werden. Dadurch muss das Passwort nicht im Klartext in der Konfigurationsdatei vorgehalten werden. Das so ausgelesene Passwort wird eventuell gesetzte Werte für `password` überschreiben.

Tipp

Wir empfehlen die Verwendung eines gesonderten Benutzerkontos.

Anmerkung

Je nach verwendeter Directory-Software und dessen Konfiguration können zur Anmeldung verschiedene Formate eines Benutzernamens zum Tragen kommen.

Neben *Down-Level Logon Name* im Stile von DOMAIN\username kann das Format auch *User Principal Name* im Stile von user@domain oder ein *Distinguished Name (DN)* wie uid=opsiconnect,cn=users,dc=test,dc=intranet sein.

Über `search_base` wird angegeben ab welchem Punkt nach passenden Element gesucht wird. Über `search_query_computers` und `search_query_groups` können Bedingungen für die Suche nach Einträgen konfiguriert werden.

Entweder `search_query_computers` oder `search_query_groups` oder beides muss konfiguriert sein. Um eine Bedingung zu deaktivieren, kann der Wert auf "" gesetzt werden. Die Suche nach Gruppen wird in einer zukünftigen Version implementiert werden. Bis dahin hat diese Einstellung keine Auswirkungen.

Der Parameter `connection_options` beinhaltet zusätzliche Optionen zur Konfiguration der Verbindung. Mit `start_tls` kann gesteuert werden, ob eine gesicherte Verbindung verwendet werden soll.

Anmerkung

Weitere Verbindungs-Optionen werden auf Nachfrage implementiert.

Ist der optionale Parameter `paged_search_limit` vorhanden und als Wert eine Ganzzahl angegeben, so werden zum Auslesen der Elemente aus dem Directory mehrere Abfragen verwendet. Wieviele Elemente eine Antwort maximal enthält wird über den gesetzten Wert gesteuert. Dieses Verhalten wird seit Version 20 unterstützt.

Über den optionalen Parameter `identifying_attribute` wird ab Version 23 festgelegt welches Attribut verwendet werden soll um einen Client eindeutig zu identifizieren. Als Standard wird hier `dn` verwendet.

Seit Version 14 ist es möglich über den Aufrufparameter `--check-directory` die Verbindungseinstellungen zum Verzeichnis zu prüfen, ohne dass eine Verbindung zum opsi-Server hergestellt wird.

Verbindung zu Univention Corporate Server

Für eine Verbindung zu Univention Corporate Server (UCS) muss für die Verbindung als Benutzername ein vollständiger *Distinguished Name* verwendet werden. Dieser hat die Form `uid=<username>,cn=users,dc=company,dc=mydomain`.

Unter UCS ist LDAP über die Ports 7389 (ungesichert) bzw. 7636 (SSL-gesichert) erreichbar. Ist auf dem Server ebenfalls Samba installiert und als AD-kompatibler Domaincontroller eingerichtet, so lauscht dieser auf den Ports 389 (ungesichert) bzw. 636 (SSL-gesichert). Für die Verwendung der SSL-gesicherten Ports muss die Verbindungseinstellung `start_tls` auf `true` gesetzt werden.

Die beiden möglichen Verbindungen unterscheiden sich auch in der Art der Anmeldung. Bei LDAP kommt `uid=...` zum Tragen, wohingegen bei Samba mittels `dn=...` gearbeitet wird.

In der Regel wird man nach Rechner-Objekten im Container `computers` suchen. Der folgende Befehl gibt den dazu passenden Wert für `search_base` aus:

```
echo "cn=computers,$(ucr get ldap/base)"
```

Für die Suche nach Windows-Clients kann `(objectClass=univentionWindows)` als Wert für `search_query_computers` angegeben werden.

Wie ein Benutzer mit nur lesendem Zugriff angelegt werden kann, ist im Univention-Wiki zu finden: [Cool Solution - LDAP search user](#)

Verhaltens-Einstellungen

Die Einstellungen steuern das Verhalten des Connectors.

```
{
  ...
  "behaviour": {
    "write_changes_to_opsi": true,
    "root_dir_in_opsi": "clientdirectory",
    "update_existing_clients": true,
    "prefer_location_from_directory": true
  },
  ...
}
```

Wird `write_changes_to_opsi` auf `false` gesetzt werden keine Daten nach opsi geschrieben. Mit dieser Einstellung ist es möglich die Verbindungseinstellungen zu überprüfen, bevor sie angewendet werden.

Per `root_dir_in_opsi` wird angegeben welche Gruppe in opsi als Wurzelgruppe verwendet werden soll. Es muss von Ihnen sichergestellt werden, dass diese Gruppe existiert.

Anmerkung

Die Gruppe `clientdirectory` wird im Configed als `DIRECTORY` angezeigt. Sollen also Clients oder Gruppen direkt unterhalb von `DIRECTORY` erscheinen, so muss als Wert für `root_dir_in_opsi` der Wert `clientdirectory` eingetragen werden.

Wird `update_existing_clients` auf `false` gesetzt, so werden bereits in opsi existierende Clients nicht verändert. Wird dieser Wert auf `true` gesetzt, so werden möglicherweise manuell gesetzte Daten mit den Werten aus dem Directory überschrieben.

Falls `prefer_location_from_directory` auf `true` gesetzt, werden Clients in opsi an die Position verschoben, welche sie im Directory haben. Für das Deaktivieren dieses Verhalten, muss dieser Wert auf `false` gesetzt werden.

Mappings

Mit einem derart flexiblen System wie ein Verzeichnisdienst benötigt der Connector Informationen darüber welche Attribute im Directory auf welche Attribute in opsi angewendet werden sollen.

```
{
  ...
  "mapping": {
    "client": {
      "id": "name",
      "description": "description",
      "notes": "",
      "hardwareAddress": "",
      "ipAddress": "",
      "inventoryNumber": "",
      "oneTimePassword": ""
    },
    "group": {
      "id": "name",
      "description": "description",
      "notes": ""
    }
  },
  ...
}
```

Es gibt jeweils ein Mapping für Clients und eines für Gruppen.

Der Schlüssel jedes Mappings ist das Attribut in opsi und der Wert ist das Attribut aus dem Verzeichnisdienst. Ist der Wert (in der Zuordnung) leer, so wird keine Zuordnung vorgenommen.

Anmerkung

Sollte der aus dem Verzeichnis ausgelesene Wert für die ID des Clients nicht als FQDN erkennbar sein, so wird ein entsprechender FQDN erstellt. Der Domain-Teil hierfür wird aus den DC-Werten des Elements gebildet.

Tipp

Unter Univention Corporate Server (UCS) kann bei `hardwareAddress` der Wert `macAddress` angegeben werden, wenn die Verbindung über LDAP (Port 7389 oder 7636) hergestellt wird.

Manuelle Zuordnung von Gruppennamen

Gruppennamen werden in der Regel ohne große Anpassungen übernommen. Allerdings kann es dabei vorkommen, dass Gruppennamen verwendet werden sollen, welche in opsi ungültig sind.

Für diese Sonderfälle kann eine manuelle Zuordnung von Gruppennamen vorgenommen werden, welche es erlaubt auch diese Fälle zu behandeln.

Zur Einrichtung wird in `mapping` ein Eintrag `group_name` angelegt, in welchem eine Zuordnung der Directory-Seite zur opsi-Seite vorgenommen wird. Für Gruppen, welche in dieser Zuordnung nicht vorkommen, wird der Namen nicht angepasst. Die Gruppennamen werden immer in Kleinbuchstaben verarbeitet, weshalb die Einträge hier in Kleinbuchstaben erfolgen müssen. Möglich ist dies ab Version 23.

Das folgende Beispiel behandelt die aus dem Directory stammende Gruppe `_server` in opsi als `server`.

```
{
  ...
  "mapping": {
    ...
    "group": {
      ...
    },
    "group_name" {
      "_server": "server"
    }
  },
  ...
}
```



Warnung

Bei unbedachtem Einsatz kann die manuelle Zuordnung unerwünschte Seiteneffekte haben. Deshalb sollte diese Zuordnungsmöglichkeit nur in Ausnahmefällen eingesetzt werden.

opsi-Verbindungs-Einstellungen

Hierüber wird gesteuert wie der Connector sich zu opsi verbindet.

```
{
  ...
  "opsi": {
    "address": "https://localhost:4447",
  }
}
```

```
    "username": "syncuser",
    "password": "secret",
    "exit_on_error": false
    "passwordFile": "",
    "connection_options": {
        "verify_certificate": true
    }
}
}
```

Unter `address` ist die Adresse des opsi-Servers einzutragen. Vergessen Sie nicht die Angabe des Ports!

Anmerkung

Ein Proxy für die Verbindung kann über die Umgebungsvariable `HTTPS_PROXY` gesetzt werden.

Mittels `username` und `password` wird geregelt welche Zugangsdaten zur Authentifizierung am opsi-Server verwendet werden. Sofern für `passwordFile` ein Wert angegeben wird, wird dieser als Pfad zu einer Datei, welche das Passwort enthält, interpretiert. Der Inhalt dieser Datei wird als Passwort verwendet werden. Dadurch muss das Passwort nicht im Klartext in der Konfigurationsdatei vorgehalten werden. Das so ausgelesene Passwort wird eventuell gesetzte Werte für `password` überschreiben.

Tipp

Wir empfehlen die Verwendung eines gesonderten Benutzers. Die Anlage zusätzlicher Benutzer ist im Dokument *Getting Started* beschrieben.

Ist der Parameter `exit_on_error` auf `true` gestellt, so führt ein Problem bei der Aktualisierung der Daten in opsi - das kann bspw. auch durch die Übermittlung von für opsi ungültige Werte geschehen - zu einem Abbruch. Steht dies auf `false`, so werden Fehler geloggt, aber der Lauf wird nicht beendet.

Unter `connection_options` werden Optionen für die Verbindung zum opsi-Server festgelegt. Mittels `verify_certificate` wird die Überprüfung des Server-Zertifikats gesteuert. Für selbstsignierte Zertifikate kann dieser Wert auf `false` gesetzt werden.

Seit Version 14 ist es möglich über den Aufrufparameter `--check-opsi` die Verbindung zum opsi-Server zu testen, ohne dass eine Verbindung zum Verzeichnisdienst hergestellt wird.

9.3.5 Den Connector ausführen

Nach der Installation existiert ein Binary `opsidirectoryconnector` auf dem System.

Dieses muss einen Parameter `--config` zusammen mit dem Pfad zur Konfigurationsdatei übergeben bekommen.

```
opsidirectoryconnector --config /etc/opsi/opsidirectoryconnector-custom.conf
```

Anmerkung

Der ausführende Benutzer benötigt keinen Zugriff auf das opsi-System, da der zugreifende Benutzer in der Konfigurationsdatei hinterlegt ist.

Beispiel: wiederkehrende Verarbeitung mit `systemd`

Der Connector macht aktuell bei der Ausführung eine Synchronisationslauf, aber die Chancen stehen gut, dass eine ständige Synchronisation erfolgt.

Es ist einfach, die Ausführung wiederkehrender Läufe zu automatisieren.

Wir werden hierbei systemd verwenden. Im Gegensatz zu cronjobs wird systemd verhindern, dass überlappende Läufe stattfinden, weshalb systemd eine gute Wahl ist.

Das folgende Beispiel wird den Connector so einrichten, dass er fünf Minuten nach dem Start der Maschine ausgeführt wird und danach jede Stunde.

Unter `/etc/systemd/system/`, dem Verzeichnis für benutzerdefinierte Units, müssen die zwei folgenden Dateien abgelegt werden. Eine Datei ist der Timer, welche unseren Job wiederkehrend aufruft und die Andere ist für den Job selbst.

Bitte füllen Sie die Datei `opsi-directory-connector.timer` mit dem folgenden Inhalt:

```
[Unit]
Description=Start the opsi-directory-connector in regular intervals

[Timer]
OnBootSec=5min
OnUnitActiveSec=1hour

[Install]
WantedBy=timers.target
```

Und dies muss nach `opsi-directory-connector.service`:

```
[Unit]
Description=Sync clients from AD to opsi.
Wants=network.target

[Service]
Type=oneshot
ExecStart=/usr/bin/opsidirectoryconnector --config /etc/opsi/opsidirectoryconnector-custom.conf
```

Um den Timer zu aktivieren und ihn sofort zu starten, können die folgenden Befehle verwendet werden:

```
systemctl enable opsi-directory-connector.timer
systemctl start opsi-directory-connector.timer
```

Falls der Timer nicht gestartet wird, wird er erst nach dem nächsten Neustart der Maschine ausgeführt werden.

Beispiel: wiederkehrende Verarbeitung als Cronjob

Es ist einfach, die Ausführung wiederkehrender Läufe über einen Cronjob zu automatisieren.

Bitte beachten Sie, dass überlappende Läufe stattfinden können, weshalb der Synchronisationsintervall am besten größer gewählt werden sollte. Zur Vermeidung dieses Problems wird die Verwendung von **systemd** anstatt **cron** empfohlen!

Zur Bearbeitung der Cronjob-Datei wird in der Regel `crontab -e` aufgerufen. Für eine zu jeder Stunde stattfindenden Synchronisation kann dort folgendes als Cronjob hinterlegt werden.

```
0 * * * * /usr/bin/opsidirectoryconnector --config /etc/opsi/opsidirectoryconnector-custom.conf
```

9.4 opsi WIM Capture

9.4.1 Vorbedingungen für die opsi Erweiterung *opsi wim capture*

Dieses Modul ist momentan eine [kofinanzierte opsi Erweiterung](#).

Es sind eine Reihe von Vorbedingungen nötig, um dieses Modul einsetzen zu können. Das bedeutet, dass Sie zum Einsatz eine Freischaltdatei benötigen. Diese Freischaltung erhalten Sie, wenn Sie die Erweiterung kaufen. Zu Evaluierungszwecken stellen wir Ihnen auch eine zeitlich befristete Freischaltung kostenlos zur Verfügung (→ mail

an info@uib.de).

Weitere Details hierzu finden Sie in Abschnitt 9.1.

Technische Voraussetzungen sind opsi 4.0.6 mit den Paketständen:

Tabelle 5: Benötigte Pakete

opsi-Paket	Version
opsi-linux-bootimage	>= 20160111
opsi-client-agent	>= 4.0.6.3-8
Windows Netboot >=7	>= 4.0.6.1-3
opsi-clonezilla	



Achtung

Für das Produkt `opsi-wim-capture` muß der share `opsi_depot_rw` für `pcpatch` beschreibbar sein. Prüfen Sie Ihre Samba Konfiguration!

- UEFI Rechner werden ab der opsi-wim-capture Version 4.1.x unterstützt
- `install.esd` (statt `install.wim`) wird als Format im Target ab der opsi-wim-capture Version 4.1.x unterstützt

9.4.2 Quick Info

Für die ganz Eiligen folgt hier eine Kurzanleitung. Ausführliche Informationen sind weiter unten zu finden.

Vorbereitungen:

- Rechner im Bios auf PXE-Boot / LAN-Boot als 1. Bootoption einstellen
- Rechner sollte mit den folgenden Property-Einstellungen im Netboot-Produkt aufgesetzt werden:


```
boot_partition_size=0
preserve_winpe_partition = true
windows_partition_label = WINDOWS
windows_partition_size=100%
```
- Target-Produkt vervollständigen:

Als Target-Produkt wird in der Regel eines der zur Verfügung gestellten Capture-Produkte verwendet, zB.: `win7-x64-capture`

Winpe- und Drivers-Verzeichnis können als symbolische Links vom Standard-Produkt erstellt werden. Das Installfiles-Verzeichnis ist zu kopieren, da hierin die `install.wim` angepasst werden soll. Zudem sollten ggfls. Dateien aus dem `custom` Verzeichnis kopiert oder verlinkt werden, zB.: `unattend.xml`
- opsi-clonezilla Property `imageshare = auto` (ist der default, veraltet `//<servername>/opsi_images`)
`imagefile` und `runcommand` werden vom `opsi-wim-capture` automatisch gesetzt
- Sämtliche Software, die in das Image integriert werden soll, muss auf dem Rechner installiert werden.

Start des Produktes opsi-wim-capture:

- Folgende Properties anpassen:


```
image_description = <Image Beschreibung>
imagename = <imagename>
target_product = win7-x64-captured
```

- opsi-wim-capture auf *setup* setzen.

Rechner mit neuem Image installieren:

- Target-Product (z.B. *win7-x64-captured*) mit folgender Anpassung auf setup setzen:
imagename = (hier den Namen aus dem gleichnamigen Property aus dem opsi-wim-capture Produkt übernehmen)

9.4.3 Einführung

Microsoft hat mit NT6 (also ab Vista) zur Installation ein neues Imageformat, das **Windows Imaging Format (WIM)** eingeführt. Ein WIM Image ist kein Platten- oder Partitionsimage, sondern mehr ein Dateien und Metadaten Archiv. Eine WIM Datei kann mehrere Images enthalten. Die normale Installation eines NT6 Rechners basiert darauf, dass die setup.exe ein Image aus der Datei install.wim auspackt, und dieses danach konfiguriert und mit zusätzlichen Treibern versieht.

Die Installation geht dadurch schneller als zu Zeiten von NT5. Leider dauern die Installationen der Hotfixes unter NT6 aber wesentlich länger, so daß eine Grundinstallation von z.B. Windows 7 zwar nur ca. eine halbe Stunde dauert, das Einspielen der Hotfixes aber etliche Stunden.

Von einem existierenden Rechner kann das Windows inclusive installierter Software, Hotfixes und Konfigurationen ausgelesen, und in Form eines WIM abgespeichert werden. Ein solches WIM kann dann wieder die Basis für neue Installationen sein.

Dazu dient das Produkt opsi-wim-capture. Im Rahmen dieses Produktes wird im Kern von der PE-Partition gebootet, und das PE liest die Systempartition aus und schreibt sie in ein WIM.

9.4.4 Abläufe Übersicht

Das Capturen eines installierten Windows läuft wie folgt ab:

Vorbereitung:

- Installation von Windows mit der Property Einstellung:
boot_partition_size=0
preserve_winpe_partition=true
windows_partition_label = WINDOWS
windows_partition_size=100%

Start des Produktes opsi-wim-capture.

Alle folgenden Punkte werden ohne weitere Interaktion vom Produkt opsi-wim-capture gesteuert:

1. opsi-clonezilla Backup der Platte (System- und winpe-Partition)
2. Backup opsi Metadaten
3. winpe Partition bootfähig machen und winpe script (work.cmd) erstellen
4. Sysprep des installierten Systems (Depersonalisierung)
5. winpe-Boot, Capture des Systems und Schreiben ins Zielprodukt
6. opsi-clonezilla Restore der Platte (System- und winpe-Partition)

9.4.5 Abläufe Details

Vorbereitung

Installation von Windows mit der Property-Einstellung: *preserve_winpe_partition=true*, da die winpe Partition später noch gebraucht wird.

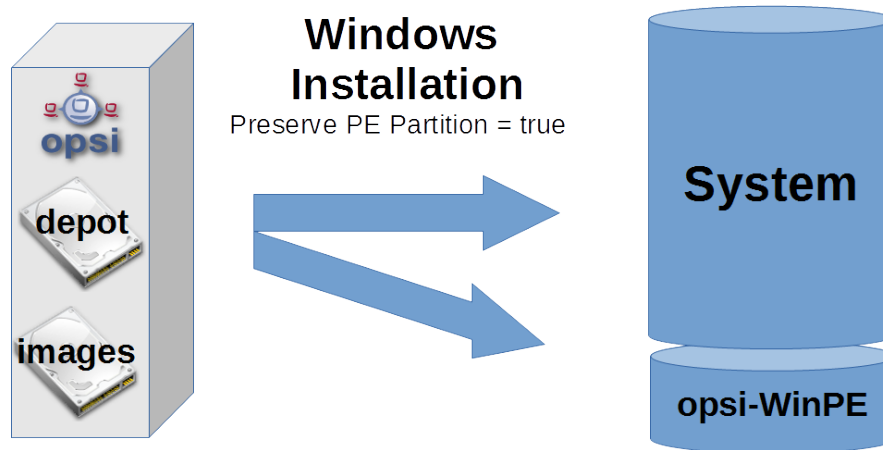


Abbildung 80: Schema: Installation des Original Windows auf der Systempartition

Nach der Windows-Installation kann nun weitere Software, Hotfixes und Konfigurationen per opsi oder händisch auf den Rechner aufgespielt werden.

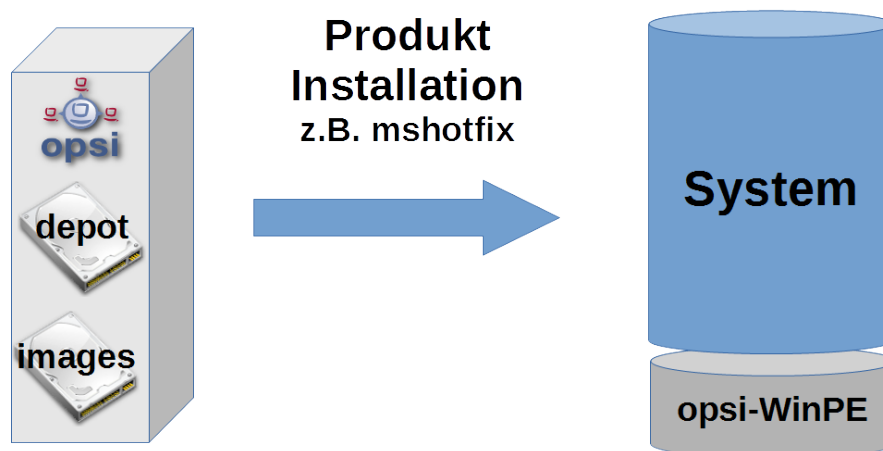


Abbildung 81: Schema: Installation von Produkten auf dem installierten System

opsi-wim-capture

Der ganze Ablauf benötigt einige Zeit. Sie sollten mit mindestens einer Stunde rechnen. Der komplette Ablauf ist aber nicht interaktiv. D.h. Sie müssen nicht dabei bleiben.

Steht das Property `disabled` auf `true` (default=false), so wird sofort abgebrochen. Dieser Schalter dient nur zu Entwicklungszwecken.

Es wird Anhand des Properties `always_backup_before_sysprep` geprüft ob ein Backup gemacht werden soll. Wenn ja, so wird über opsi-clonezilla ein Plattenbackup ausgelöst.

Anmerkung

Für opsi-clonezilla wird das runcommand:

`ocs-sr -q2 --batch -j2 -rm-win-swap-hib -i 2000 -p true savedisk imagefile sda` gesetzt. Innerhalb diese Kommandos wird `imagefile` abhängig von dem Wert des Properties `clonezilla_imagefile` gesetzt: Steht diese Property auf `auto` (default), so wird der Wert für `imagefile` automatisch erstellt. Dies geschieht unter Verwendung von Propertywerten und dem Clientnamen nach dem Muster:

`<FQDN des Clients>_<target_product>_<imagename>`

Bei einem anderen Wert als `auto` wird der angegebene Wert als `Imagefile` verwendet. Weiterhin wird das Produkt opsi-clonezilla auf `setup` gesetzt. Damit das Produkt opsi-clonezilla startet ist nun ein Reboot nötig.

Um eine Endlosschleife zu vermeiden, wird nun ein Rebootflag gesetzt, damit nach Beendigung des Backup erkannt werden kann, dass dieser Schritt bereits erledigt ist.

Technischer Hinweis: Hier entsteht das Problem, dass der Rebootflag auch in dem Backup landet, aber nach einem Restore nicht mehr erwünscht ist. Daher wird der Rebootflag als Timestamp gesetzt. Ein Rebootflag, der älter als 0,1 Tage (=2,4 Stunden) ist wird ignoriert.

Die Maschine wird rebootet, dabei bleibt das Produkt `opsi-wim-capture` auf `setup` stehen. Nun startet das Netboot Produkt opsi-clonezilla und führt das Backup aus.

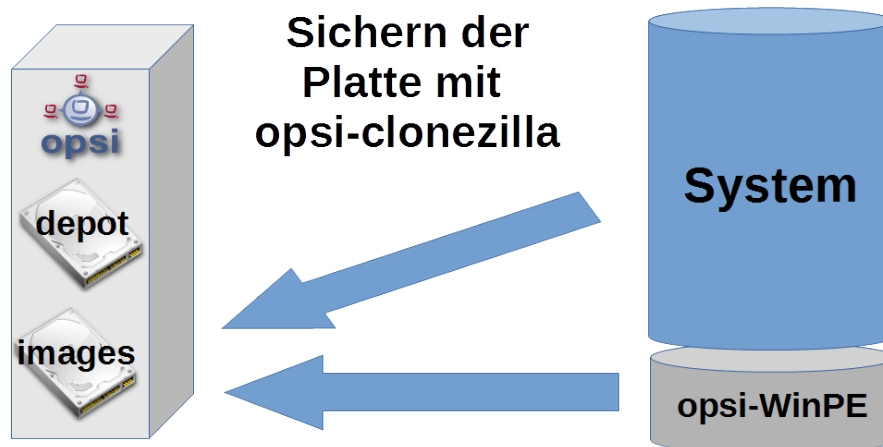


Abbildung 82: Schema: Backup der Platte mit opsi-clonezilla

Tipp

Warum opsi-clonezilla Backup ?

Das nachfolgende Sysprep macht die Systempartition für die weitere Verwendung unbrauchbar.

Ein vom erstellten (captured) WIM-Image erstelltes System enthält Informationen über das gelaufene Sysprep und ist nicht als Basis für weitere opsi-wim-capture Läufe geeignet.

Erneutes capturen immer auf Basis des per restore wiederhergestellten opsi-clonezilla Images ausführen.

Das Produkt opsi-clonezilla wird nun so eingestellt, das ein erneuter start ein Restore durchführen wird.

Tipp

Muß ein opsi-clonezilla Backup erstellt werden?

Wenn der Rechner lediglich zum Erstellen eines WIM-Captures dient und danach neu Installiert wird, oder es sich um einen virtuellen Rechner handelt der sich aus einem Snapshot wieder herstellen läßt, kann auf die Erstellung eines Backups mit Clonezilla und den abschließenden Restore verzichtet werden.

Die entsprechenden Properties sind `always_backup_before_sysprep` und `start_after_capture`.

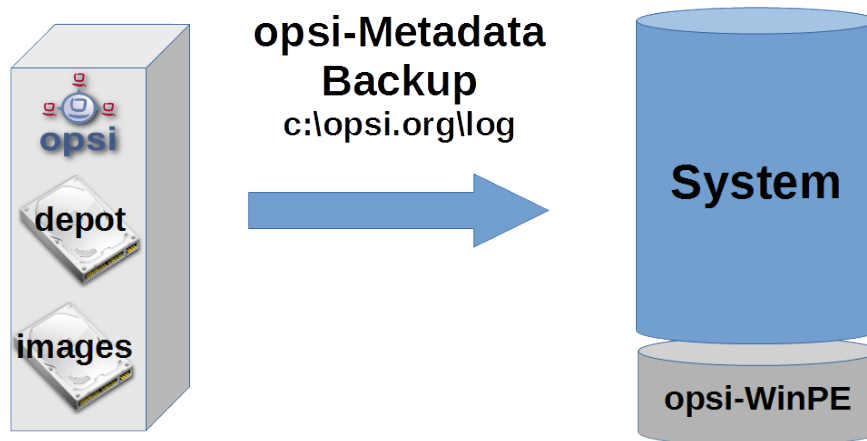


Abbildung 83: Schema: Sicherung der opsi-meta-daten nach c:\opsi.org\tmp

Nun werden die opsi Informationen, welche opsi-Produkte in welcher Version auf dem Client installiert sind, auf dem Client hinterlegt.

Anmerkung

Die productOnClient Objekte für alle Localboot Produkte werden nach c:\opsi.org\tmp\productonclients.json geschrieben.

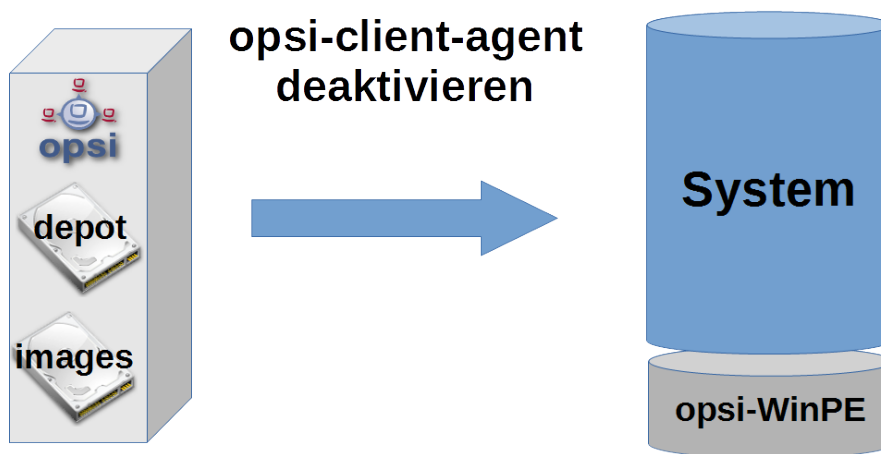


Abbildung 84: Schema: Deaktivierung des opsi-client-agenten

Der opsi-client-agent des Rechners wird deaktiviert, damit er beim späteren Ausrollen des Images nicht aktiv wird.

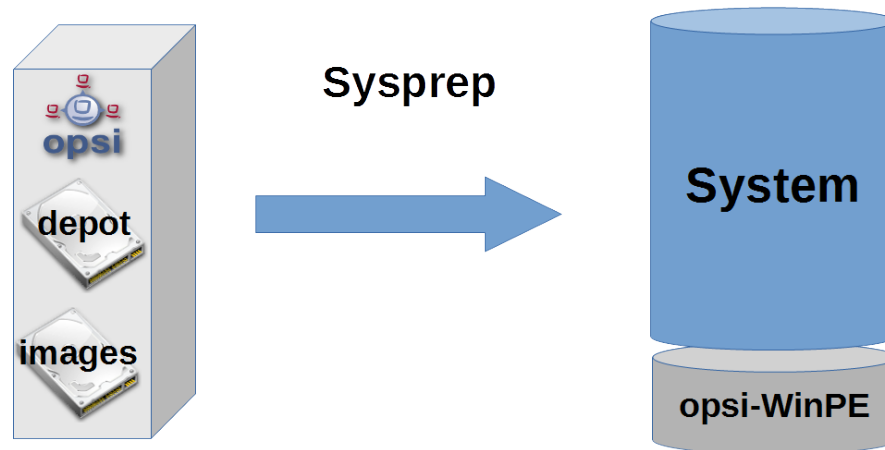


Abbildung 85: Schema: Depersonalisierung der Systempartition mit *sysprep*

Damit das Image, welches erstellt werden soll, sich wie ein Standard Windows Setup auf einem beliebigen Rechner ausrollen läßt, muß es depersonalisiert werden. Dies wird mit dem Windows Werkzeug *sysprep* erledigt.

Tipp

Installierte Software wird nicht depersonalisiert. Es ist durchaus möglich, dass installierte Software sich in Ihrer Konfiguration merkt, auf welchem Rechner sie installiert wurde. Eine solche Konfiguration wird dann wahrscheinlich Probleme machen, wenn das Image auf einem anderen Rechner ausgerollt wird. Von daher ist es nicht die ideale Idee, möglichst viel Software in einem Image unterzubringen.

Steht das Property `startcapture` auf *false* (default=true), so wird die Arbeit nach dem *sysprep* abgebrochen und der Rechner heruntergefahren. Dies ist nur sinnvoll wenn von dem Rechner danach mit einem anderen Werkzeug ein Image erstellt werden soll.

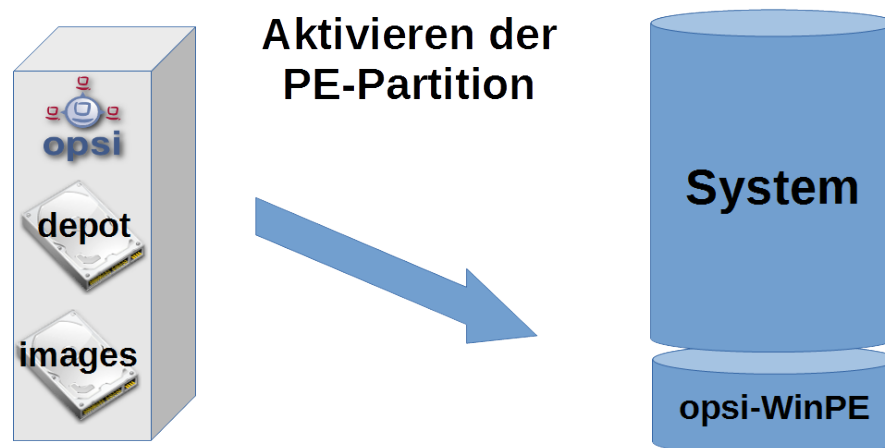


Abbildung 86: Schema: Aktivieren und bootbar machen der PE Partition

Das Auslesen der Windows-Partition und Wegschreiben in die WIM-Datei muss von einem Windows erfolgen, welches nicht das Windows ist, welches gelesen werden soll. Vielmehr wird hierfür das Windows PE verwendet, welches bei der ursprünglichen Installation angelegt und *aufgehoben* wurde.

- Aktivierung des WinPE als bootbare Partition, Erstellung der nötigen Bootrecords und, soweit nötig, Deaktivierung von Laufwerksbuchstaben bei anderen Partitionen.

- Auslesen der opsi-Metadaten über installierte Produkte auf dem Client und Speicherung dieser Daten auf dem Client in einem temporären Verzeichnis.
- Einige Aufräumarbeiten auf dem auszulesenden System.

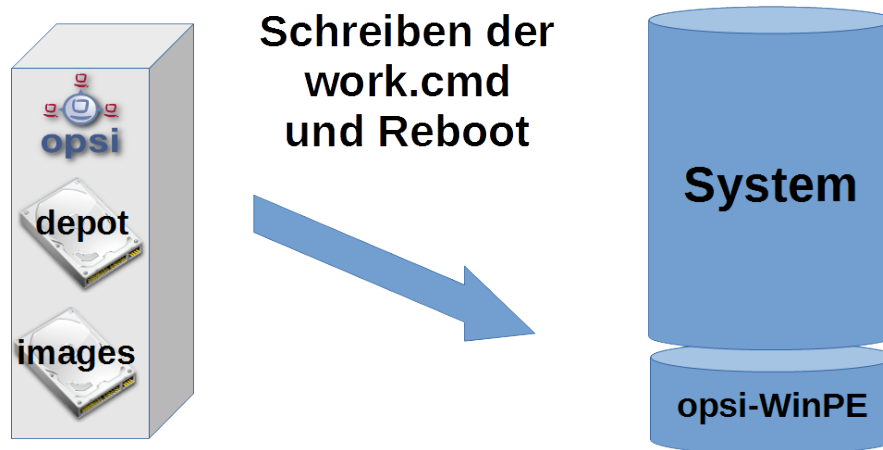


Abbildung 87: Schema: Erstellen der work.cmd im PE

- Schreiben einer Kommandodatei, welche die Capturevorgänge beim nächsten WinPE-Start initiiert.
- Bereitstellen weiterer Daten für die Abläufe im WinPE, wie z.B. Liste der Produkte aus dem Property `start_after_capture`
- Reboot des Clients

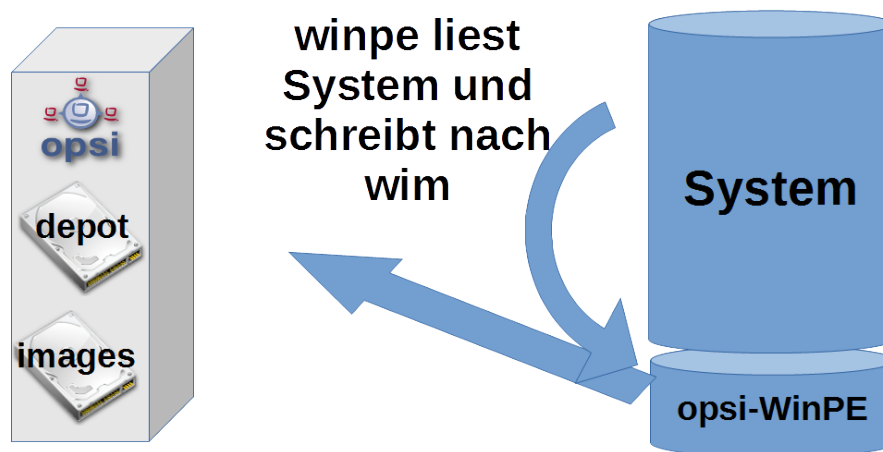


Abbildung 88: Schema: Capture der Systempartition vom PE aus

In dieser Phase startet das WinPE und führt nun den eigentlichen Capturevorgang durch. Im Detail:

- Mounten des `opsi_depot_rw` shares, damit auf diesen auch geschrieben werden kann.
- Prüfen der Architektur des WinPE (32/64 Bit) und Start des opsi-script in der entsprechenden Architektur.
- Herstellung der Verbindung zum opsi-webservice
- Reaktivierung der Laufwerksbuchstaben

- Wenn das Property `check_disk_before_capture` den Wert `true` hat (default=false) dann wird nun ein `chkdsk` für die Windows Partition ausgeführt. Dies dauert lange.
- Es wird geprüft ob das über das Property `target_product` angegebene Produkt auf dem Share `opsi_depot_rw` existiert und eine `install.wim` Datei an der richtigen Stelle besitzt.
- Prüfen und Erstellen einer Lock-Datei im `target_product`. Wenn diese Datei bereits existiert, so wird hier abgebrochen, um zu vermeiden, dass mehrere capture Vorgänge gleichzeitig in die selbe WIM-Datei schreiben.
- Wenn das Property `force_imagex` den Wert `true` hat (default=true), dann wird das `imagex` Programm des Produktes `opsi-wim-capture` zum Capturen verwendet, auch wenn das Windows PE über das Programm `dism` verfügt. Ansonsten wird `dism` verwendet, wenn verfügbar. `Dism` ist schneller, kann aber zu Images führen, welche sich nicht ausrollen lassen.
- Wenn das Property `capture_mode` den Wert `append` hat: Überprüfen, ob ein Image mit diesem Namen in der `install.wim` schon vorhanden ist, und gegebenenfalls dieses löschen. Der Wert `always_create` wird nur akzeptiert, wenn als Werkzeug `dism` verwendet wird. In diesem Fall wird eine neue `install.wim` Datei erzeugt.
- Start des Capturevorgangs. Hierzu wird das weiter oben ausgewählte Werkzeug (`imagex` oder `dism`) und der ausgewählte `capture_mode` verwendet. Der Name des Images wird durch das Property `imagename` festgelegt. Die hinterlegte Beschreibung des Images wird durch das Property `image_description` festgelegt. Dies kann lange dauern.



Achtung

Imagename merken! Der Name des erstellten Images wird momentan noch nicht automatisch in die Liste der möglichen Imagennamen aufgenommen. Sie müssen sich den Namen merken und beim Ausrollen angeben!

- Löschen der Lock Datei im `target_product`.
- Die entstandenen Logfiles werden zusammengeführt.
- Setzen der Produkte aus dem Property `setup_after_capture` auf `setup`. Dabei werden auch die Produktabhängigkeiten der betroffenen Produkte aufgelöst. Dieses Property ist eine Liste und kann auch mehr ProduktIds aufnehmen.

Tipp

opsi-clonezilla auf `setup` stellen lassen!

Der Rechner ist nach dem Capture Vorgang depersonalisiert und damit weitgehend unbrauchbar. Das Produkt opsi-clonezilla ist so vorbereitet, dass ein weiter oben erstelltes Backup automatisch wieder hergestellt wird, wenn es hier auf `setup` gestellt wird.

- Deaktivierung der WinPE Partition und Aktivierung der Systempartition (Windows).
- Schreiben der Logdatei zum Server. Dort wird diese an die Logdatei des opsi-wim-capture Laufs angehängt.
- Reboot

Wenn das Produkt `opsi-clonezilla` hier auf `setup` gestellt worden ist, so wird nun automatisch ein Restore der Platte durchgeführt.

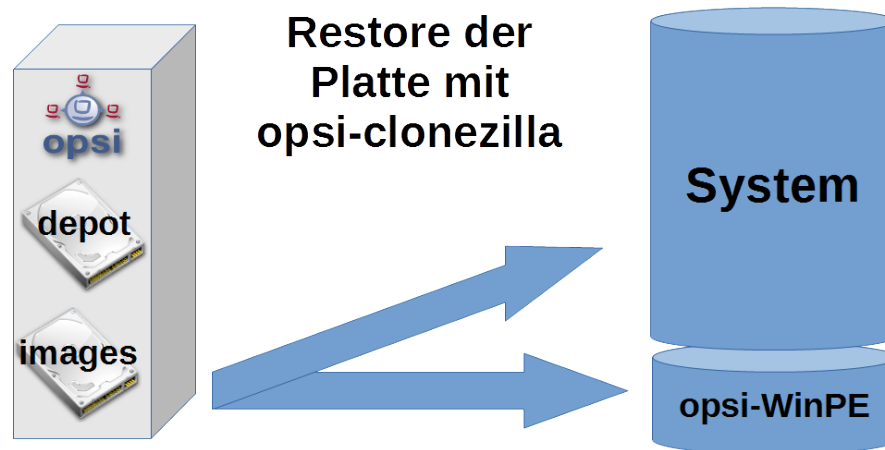


Abbildung 89: Schema: Restore mit opsi-clonezilla

9.4.6 Produkte

Hauptprodukt opsi-wim-capture

Das Produkt opsi-wim-capture hat folgende Produktproperties:

- **always_backup_before_sysprep:**
(true/false), Default=true,
Startet immer ein opsi-clonezilla Backup vor dem sysprep Vorgang.
- **startcapture:**
(true/false), Default=true,
Startet nach dem Sysprep den Capture Prozess und rebootet den Rechner. Wenn false wird nach dem Sysprep der Rechner herunter gefahren.
- **disabled:**
(true/false), Default=false,
Wenn true wird das Produkt nicht ausgeführt. Dieses Property wird normalerweise nicht benötigt und dient nur zu Debugzwecken.
- **target_product:**
Name des Ziel Produktes (Default = ")



Wichtig

Dieses Property ist nicht *schlau*, d.h. es wird nicht überprüft, ob das ausgelesene Image zum Zielprodukt passt. Sie können also ohne Fehlermeldung ein win7-32Bit Image in ein Win81-64Bit Produkt schreiben. Das sollten Sie aber nicht! Wir empfehlen die Verwendung von gesonderten Produkten, welche nur als Ziel dienen (z.B. win7-x64-captured).

Das Zielprodukt muß, genauso wie ein normales Produkt, zur Windows Installation vorbereitet werden. Als Zieldatei innerhalb des Zielproduktes dient die `install.wim` Datei (`installfiles/sources/install.wim`), welche auch die von Microsoft gelieferten Images enthält. Ob das ausgelesene Image nun an diese Datei angehängt werden soll, oder eine neue `install.wim` erzeugt werden soll, steuert das Property:

- **capture_mode:**
(append/always_create) Default=*append*:

Bei `append` wird das neu erstellte Image an die vorhandene `install.wim` angehängt.



Wichtig

Enthält die `install.wim` schon ein Image gleichen Namens wird dieses **ohne Nachfrage gelöscht**.
Bei `always_create` wird eine neue `install.wim` erstellt.
`always_create` funktioniert nicht mit WinPE-Installationen, die auf Windows < 8 basieren.

Die `Install.wim`-Datei ist ein Container, der mehrere Images enthalten kann. Die Images haben einen Namen und eine Beschreibung. Der Name und die Beschreibung des neu erstellten Images werden durch die folgenden Properties gesteuert:

- `imagename`:
Default = "
- `image_description`:
Default = "
- Das Property `start_after_capture`
ist ein Liste von Produkten, welche nach dem Abschluß des Capturevorgangs auf `setup` gestellt werden sollen. Eine gute Idee ist hier zum Beispiel `opsi-clonezilla`, welches das vor dem `sysprep` erstellte Backup wiederherstellt.
- `force_imagex`:
`true/false` (default=`true`) Soll für den capture-Vorgang das Werkzeug `imagex` verwendet werden, auch wenn im WinPE das Werkzeug `dism` zur Verfügung steht.
- `opsi_depot_rw_host`:
Normalerweise `auto` (default) oder leer lassen.
Wenn nicht `auto` oder leer: der Host von dem der share `opsi_depot_rw` gemountet werden soll. Wenn der Host angegeben wird, dann als Hostname, FQDN oder IP-Nummer.
Diese Property dient nur für Fälle bei denen der share `opsi_depot_rw` **nicht** über das Depot dem der Client zugewiesen ist erreichbar ist.
- `checkdisk_before_capture`:
Soll for dem capture Vorgang ein file system check der Systempartition durchgeführt werden.
Default = `false`.
- `verify_clonezilla_images`:
Soll Clonezilla die Images auf Lesbarkeit überprüfen: `after_save`, `before_restore`, `never`, `always`
Eine Überprüfung dauert in etwa genauso lang wie der Schreib- oder Leseprozess.
Default = `never`

Target Produkte

Die Target Produkte dienen dazu die gecapturten Images aufzunehmen.

Warum Target-Produkte ?

Die Target-Produkte unterscheiden sich nicht von den Standard opsi Windows-Install Produkten. Technisch kann also z.B. ein normales `win7-x64` als TargetProdukt dienen.

Wir empfehlen die Verwendung von Target Produkten, um eine Installation aus dem unmodifizierten Abbild einer original Microsoft DVD von einer Installation, welche aus einer modifizierten `install.wim` kommt, abgrenzen zu können. Weiterhin haben Sie somit noch ein Produkt in *Reserve*, sollte bei einem capture mal die `install.wim` unbrauchbar werden. Die Entscheidung, was Sie als Target Produkt verwenden, liegt natürlich bei Ihnen.

Wir liefern die folgenden Target Produkte aus:

- `win7-x64-captured`

- win81-x64-captured
- win10-x64-captured

Sie müssen diese Produkte genauso *befüllen*, wie andere Windows Netboot Produkte (siehe hierzu opsi-getting-started Handbuch).

Dabei dürfen Verzeichnisse wie *winpe* oder z.B. *drivers/drivers/additional/byAudit* durchaus symbolische Links auf die entsprechenden Verzeichnisse aus dem passenden Nicht-Target-Produkt sein. Achtung: das *installfiles* Verzeichnis muß tatsächlich mit dem Inhalt der Windows DVD befüllt werden und darf kein symbolischer Link sein.

9.4.7 Windows Installation von einem Targetprodukt aus

(Ausrollen des gecapturten Images)

Wiederherstellung der opsi Metadaten zu installierten Produkten

Das Problem:

Wenn Sie ein Windows mit opsi neu installieren, z.B. aus *win7-x64*, dann werden bei der Installation des opsi-client-agent alle Localboot-Produkte, welche bei diesem Rechner vorher auf *installed* standen, automatisch auf *setup* gestellt und damit später erneut installiert.

Dies kann beim Ausrollen eines *gecapturten* Images nicht ganz genauso durchgeführt werden.

Im Image befindet sich das Backup der opsi-Daten, das dort während des capture Vorgangs abgelegt wurde. Dieses wird bei der Installation des opsi-client-agent entdeckt, und wieder in den opsi-server eingespielt. Damit stehen die Produkte, die in dem *gecapturten* Image installiert waren, jetzt für den frisch installierten Rechner auf *installed*. Würden jetzt alle Produkte, welche auf *installed* stehen auf *setup* gesetzt, würde dies dazu führen, dass alle schon im Image installierten Produkte nochmal installiert werden. Dies ist nicht erwünscht.

Bei der Wiederherstellung der opsi Metadaten zu installierten Produkten gibt es ab opsi 4.0.7 zwei Varianten:

- Variante 1:
Zurückspielen der Metadaten und Beibehaltung von *setup*-Actionrequests.
Produkte die auf *installed* stehen werden **nicht** auf *setup* gestellt.
Dies ist der Default und das Verhalten vor opsi 4.0.7
- Variante 2:
Zurückspielen der Metadaten. Produkte die auf *installed* stehen werden auf *setup* gestellt ausser denen welche in den restorten Metadaten enthalten waren.

Variante 1

Beim Ausrollen eines *gecapturten* Images werden nach der Installation des Images nur die Produkte automatisch installiert, welche schon vor dem Beginn der Betriebssystem-Installation auf *setup* standen. Dies kann durch Ihren Eingriff oder das Property *setup_after_install* erfolgt sein. Daher werden in diesem Fall auch nur die Produkte installiert, welche vor der Installation des Betriebssystems auf *setup* standen.

Dies ist der Default und das Verhalten vor opsi 4.0.7

Variante 2

Die Variante 2 verhält sich vom Ergebnis ähnlich wie es bei Installationen aus nicht gecapturten Images der Fall ist:

* Zurückspielen der Metadaten.

* Produkte die auf *installed* stehen werden auf *setup* gestellt ausser denen welche in den restorten Metadaten enthalten waren.

Dieses Verhalten steht erst ab opsi 4.0.7 zur Verfügung und ist nicht der Default. Variante 2 ist durch Erweiterungen am opsi-script möglich geworden und ist Bestandteil des opsi-client-agent von 4.0.7.

Um dieses Verhalten zu verwenden muss ein *config* (*Hostparameter*) gesetzt werden:

Der boolsche Konfigurationseintrag: *clientconfig.capture.switch_installed_products_to_setup*. Hat dieser Eintrag für den Client den Wert *true* dann wird Variante 2 verwendet, ansonsten Variante 1.

Über diese *Hostparameter* können dann Events Client-spezifisch aktiviert bzw. deaktiviert werden. Die *Hostparameter* können über den *opsi-configed* oder *opsi-admin* angelegt werden.

Zum Anlegen der *Hostparameter* über *opsi-admin* sind die folgenden Befehle auf dem *opsi-configserver* auszuführen:

```
opsi-admin -d method config_createBool clientconfig.capture.switch_installed_products_to_setup "capture.\
switch_installed_products_to_setup" true
```

Damit stellen Sie für **alle** Rechner *Variante 2* ein.

Zum Anlegen der *Hostparameter* über den *opsi-configed* wählen Sie dort *Serverkonfiguration* / *clientconfig* / Auf der Rechten Seite mit der rechten Maustaste: **Boolschen Konfigurationseintrag hinzufügen**.

9.4.8 Hilfsprodukt opsi-wim-info

Das Produkt *opsi-wim-info* kann verwendet werden um schnell Informationen über die in einer *install.wim* gespeicherten Images auszulesen. Diese Informationen werden dann in der Logdatei gespeichert.

Properties:

- **target_produkt**
ProductId des Produktes in dem die *install.wim* gesucht wird.

9.4.9 Bekannte Einschränkungen und Probleme

Folgende Einschränkungen sind derzeit (13.7.2018) bekannt:

- keine

9.5 opsi Linux Support








































































9.5.1 Unterstützt als opsi-client: Linux

(Stand / as of 26.09.2018)

Tabelle 6: Supported Linux OS as Client in opsi 4.1 and 4.0.7 /
Unterstützte Linux-OS als Client in opsi 4.1 und 4.0.7

Distribution	OS-Installation	netboot products	client-agent	opsiclientd
Debian 9 <i>Stretch</i>	✓	debian, debian9	✓	✓
Debian 8 <i>Jessie</i>	✓	debian, debian8	✓	✓
Debian 7 <i>Wheezy</i>	⚠	debian, debian7	⚠	⚠
Debian 6 <i>Squeeze</i>	⚠			
Ubuntu Bionic 18.04 LTS	✓	ubuntu, ubuntu18-04	✓	✓
Ubuntu Xenial 16.04 LTS	✓	ubuntu, ubuntu16-04	✓	✓
Ubuntu Wily 15.10	⚠	ubuntu, ubuntu15-10	⚠	✗
Ubuntu Vivid 15.04	⚠	ubuntu, ubuntu15-04	⚠	✗
Ubuntu Utopic 14.10	⚠	ubuntu	⚠	✗
Ubuntu Trusty 14.04 LTS	✓	ubuntu, ubuntu14-04	✓	✓

Tabelle 6: (continued)

Ubuntu Precise 12.04 LTS		ubuntu		
Ubuntu Lucid 10.04 LTS				
RHEL 7		rhel70		
RHEL 6				
CentOS 7		centos70		
CentOS 6				
SLES 15				
SLES 12.3				
SLES 12.2		sles12sp2		
SLES 12.1		sles12sp1		
SLES 12		sles12		
SLES 11SP4		sles11sp4		
SLES 11SP3		sles11sp3		
openSuse Leap 15.0		opensusel15		
openSuse Leap 42.3		opensusel42-3		
openSuse Leap 42.2		opensusel42-2		
openSuse Leap 42.1		opensusel42-1		
openSuse 13.2		opensuse13-2		
openSuse 13.1 RC2				
openSUSE 12.3				
openSuse Tumbleweed				
UCS 4.3		ucs43		
UCS 4.2		ucs42		
UCS 4.1		ucs41		
UCS 4.0				
UCS 3.2				
UCS 3.0				

 : Supported  : Unsupported  : Under Development  : Discontinued

Tabelle 7: Linux netboot products and the used installer type in opsi 4.1 and 4.0.7 / Linux Netboot-Produkte nach Installer-Typ in opsi 4.1 und 4.0.7



Netbootproduct	Installer	State	Remark
debian	opsi		squeeze - stretch
debian9	distribution		

Tabelle 7: (continued)

debian8	distribution	✓	
debian8	distribution	✓	
debian7	distribution	⚠	
ubuntu	opsi	✓	trusty - artful
ubuntu18-04	distribution	✓	
ubuntu16-04	distribution	✓	
ubuntu15-10	distribution	⚠	
ubuntu15-04	distribution	⚠	
ubuntu14-04	distribution	✓	
centos70	distribution	✓	
redhat70	distribution	✓	
sles15	distribution	🚧	
sles12sp2	distribution	✓	
sles12sp1	distribution	✓	
sles12	distribution	✓	
sles11sp4	distribution	✓	
sles11sp3	opsi	⚠	
opensusel15	distribution	🚧	
opensusel42-3	distribution	✓	
opensusel42-2	distribution	⚠	
opensusel42-1	distribution	⚠	
opensuse13-2	distribution	⚠	
opensuse13-1	opsi	⚠	
ucs43	distribution	✓	
ucs42	distribution	✓	
ucs41	distribution	⚠	

9.5.2 Vorbedingungen für den opsi Linux Support

Technische Voraussetzungen sind opsi 4.0.5 mit den Paketständen:

Tabelle 8: Benötigte Pakete

opsi-Paket	Version
opsi-linux-bootimage	>= 20140805-1

Der opsi Support für Linux besteht aus einem Teil der von Anfang an Opensource ist (den Netbootprodukten) und einem kofinanzierten Teil (dem Agent für die Clients).

Dieser opsi-linux-client-agent ist eine [kofinanzierte opsi Erweiterung](#).

Das bedeutet, dass Sie zum Einsatz eine Freischaltdatei benötigen. Diese Freischaltung erhalten Sie wenn Sie die Erweiterung kaufen. Zu Evaluierungszwecken stellen wir Ihnen auch eine zeitlich befristete Freischaltung kostenlos zur Verfügung (→ mail an info@uib.de).

Weitere Details hierzu finden Sie in Abschnitt [9.1](#).

9.5.3 opsi-linux-client-agent: 15 Freistarts

Seit opsi 4.0.7 beinhaltet der opsi-linux-client-agent 15 Freistarts bei denen der Agent auch ohne Freischaltung verwendet werden kann.

Genauer formuliert: Nach der initialen Installation des opsi-linux-client-agent kann der der opsi-script 15 mal im Servicekontext gestartet werden ohne eine Freischaltung zu fordern.

Dies gibt Ihnen die Möglichkeit einen Linuxrechner aufzusetzen und mit den entsprechenden opsi-Produkten für den geplanten Einsatz zu konfigurieren. Beispielsweise können Sie nach der Installation das Produkt `1-opsi-server` aufrufen um aus dem frisch installierten Rechner einen opsi-server zu machen.

Für eine dauerhafte Pflege des installierten Linuxrechners über diese 15 Freistarts hinaus benötigen Sie aber eine Freischaltung dieses Features.

9.5.4 Einspielen der Produkte

Die Linux bezogenen Localboot und Netbootprodukte können über den opsi-product-updater geladen werden. Dazu muss sichergestellt sein, dass in der `/etc/opsi/opsi-product-updater.conf` auch die entsprechenden Verzeichnisse eingetragen sind. Am einfachsten ist es, wenn sich folgende Sektion in der Konfigurationsdatei findet:

```
[repository_uib_linux]
active = true
baseUrl = http://download.uib.de
dirs = opsi4.0/products/opsi-linux
autoInstall = false
autoUpdate = true
autoSetup = false
; Set Proxy handler like: http://10.10.10.1:8080
proxy =
```

Dann können Sie mit dem folgenden Aufruf die opsi-linux Produkte einspielen:

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

9.5.5 Einführung

Ein Management-Werkzeug für Windows und Linux

Ziel der Erweiterung von opsi um die Unterstützung von Linux-Systemen ist die Schaffung eines Managementsystems für heterogene Umgebungen. Der Fokus liegt dabei auf der möglichst vollständigen Integration beider Welten in die gleichen Management-Vorgänge und Werkzeuge.

Dies bedeutet, dass eine Linux-Installation auf die gleiche Weise angestoßen wird wie eine Windows-Installation. Der opsi-client-agent unter Linux basiert auf dem selben Code wie der unter Windows und ist (soweit sinnvoll) befehlskompatibel.

Linux-Distributionsübergreifend

Der Linux-Support von opsi ist distributionsübergreifend angelegt. Die folgenden Distributionen werden gleichwertig unterstützt:

- Debian
- Ubuntu
- OpenSuse / SLES (Suse Linux Enterprise Server)
- RHEL (RedHat Enterprise Linux)
- CentOS
- UCS

9.5.6 Linux Netboot Produkte v4.0.6 auf Basis des Distributionseigenen Installers

Bei den mit opsi v4.0.5 veröffentlichten Linux-Netboot-Produkte wurden die Installationen des gewählten Betriebssystems weitgehend vom Netboot-Produkt gesteuert. Die entsprechenden Produkte setzen ab v4.0.6 auf den distributionseigenen Installer.

Dies ist ein grundsätzlicher Umbau, der dazu führt, das sowohl Aufbau als auch Verhalten der Produkte unterschiedlich zu den bisherigen Produkten sind.

Hier eine Übersicht:

- Ähnlich wie bei der Windows-Installation wird für den Installer eine Antwortdatei bereit gestellt, welche vom Installer zur nichtinteraktiven Installation genutzt wird.
- Der distributionseigene Installer ist nicht wie bei Windows ein Programm das aufgerufen wird, sondern in einer Kombination aus distributionseigenem Kernel und initrd implementiert.
- Die gesamte Grundinstallation - inklusive Partitionierung, LVM, Basissoftware, etc. - liegt in der Hand des Installers und wird nicht mehr durch das bootimage durchgeführt.
- Bei den Suse- und RedHat-artigen Distributionen werden die Installationsquellen von Ihnen bereitgestellt, in dem Sie die Installations-DVD als ISO-Datei auf dem Depotshare ablegen. Dieses Verfahren ähnelt der Situation unter Windows, nur dass der Ablageort ein anderer ist und dass Sie bei Windows den Inhalt der Installations-DVD ablegen anstatt einer ISO-Datei.
- Bei den Debian-artigen werden die Installationsquellen aus dem Netz verwendet. Auf dem Depotshare liegen nur die Netboot Versionen von Distributionskernel und dazugehörigem initrd. Da diese Dateien nicht groß sind, werden sie im opsi-Paket mitgeliefert.
Ab opsi 4.0.7 können für bestimmte Netbootprodukte hier auch lokale http Repositories bereit gestellt werden.
- Zur weiteren Pflege der Installation kann der opsi-linux-client-agent im Rahmen der Basisinstallation mit installiert werden.

Abläufe der neuen Installationsmimik:

1. Das opsi-linux-bootimage wird gebootet, löscht die Partitionstabelle und erstellt eine kleine temporäre Hilfspartition.
2. Das opsi-linux-bootimage holt sich das distributionseigene initrd und entpackt es auf der Hilfspartition.
3. Das opsi-linux-bootimage holt sich die generische Vorlage für die Antwortdatei, patcht (personalisiert) diese und legt sie dann in das initrd Verzeichnis.
4. Das opsi-linux-bootimage erstellt weitere Hilfscripte und Konfigurationsdateien (z.B. zur Installation des opsi-linux-client-agent) und legt sie dann in das initrd Verzeichnis.
5. Das opsi-linux-bootimage packt das gepatchte initrd Verzeichnis wieder zusammen.
6. Das opsi-linux-bootimage bootet den Distributions-Kernel mit dem gepatchten initrd per kexec.

7. Das so geladene System installiert das Zielsystem unattended und installiert abschließend den opsi-linux-client-agent.

Die Vorteile dieses Vorgehens sind:

- Die Installation findet exakt gemäß den Anforderungen des Distributors statt. Dies ist immer ein Vorteil, aber natürlich im Unternehmensumfeld als Ausgangsbedingung für Supportverträge besonders wichtig.
- Die Integration neuer Releases in opsi wird einfacher und dadurch schneller.
- Bei den Suse- und RedHat-artigen Distributionen findet die Installation aus auf dem opsi-Server liegenden Installationsquellen statt und ist damit schneller und unempfindlicher gegen Störungen als beim Zugriff auf Repositories aus dem Internet.

Es ergeben sich aber auch folgende Nachteile:

- Im Moment noch keine Unterstützung für UEFI-Rechner. Die Informationen über die UEFI-Umgebung gehen aktuell beim Boot per kexec verloren. Wir hoffen, dass wir dieses Problem in Zukunft mit einem neueren Bootimage beheben können.

Bereitstellung und der Installationsmedien auf dem Server

Die Bereitstellung der Installationsmedien für die Suse- und RedHat-artigen Distributionen erfolgt auf einem nfs share: `opsi_nfs_share`.

Zur Einrichtung des shares muss ein NFS-Server auf dem opsi-server installiert und konfiguriert sein.

Seit opsi v4.0.6 wird dies über ein gesondertes Paket `opsi-linux-support` erfolgen. Dieses Paket wird nicht per default installiert und muss einmalig nachinstalliert werden.

Auf Debian-artigen Betriebssystemen kann das durch den folgenden Befehl erreicht werden:

```
apt-get install opsi-linux-support
```

Beim Einsatz einer Firewall auf Ihrem Server muss diese noch so konfiguriert werden, dass TCP-Verbindungen auf Port 80 akzeptiert werden. Bitte konsultieren Sie hierzu das entsprechende Handbuch.

Was dieses Paket macht ist (als händige Anleitung) im folgenden Beschrieben:

- Auf dem opsi-server muß das entsprechende NFS-Server-Paket installiert sein. Auf Debian, Ubuntu, Suse ist dies das Paket: `nfs-kernel-server`. Auf Centos, Redhat ist es das Paket `nfs-utils`.
- Der Export `opsi_nfs_share` muß angelegt und exportiert werden:
 - Verzeichnis erzeugen:
`mkdir -p /var/lib/opsi/depot/opsi_nfs_share`
 - In der Datei `/etc/exports` den Eintrag:
`/var/lib/opsi/depot/opsi_nfs_share *(ro,no_root_squash,insecure,async,subtree_check)`
erzeugen.
 - Das Aktivieren des Exports wird mit dem folgenden Befehl ausgelöst:
`exportfs -r`
 - Zur Kontrolle des erfolgreichen Exports den folgenden Befehl aufrufen:
`showmount -e localhost`
Die Ausgabe sollte sein:
`Export list for localhost: + /var/lib/opsi/depot/opsi_nfs_share *`

- Der share `opsi_nfs_share` hat folgenden Verzeichnisaufbau:
`opsi_nfs_share/<productId>/<arch>/<dvd>.iso`
zum Beispiel:
`opsi_nfs_share/opensuse13-2/64/openSUSE-13.2-DVD-x86_64.iso`
Die Installationsdatei muß als Dateiendung `.iso` haben, der Rest ist egal. Liegen in einem Verzeichnis mehrere `.iso` Dateien so ist nicht definiert welche verwendet wird.
- Kopieren Sie die Installations-DVD an den entsprechenden Platz im `opsi_nfs_share` und führen Sie aus:
`opsi-set-rights /var/lib/opsi/depot/opsi_nfs_share`
WICHTIG: Verwenden Sie die Standard Installations-DVD's der Distribution. Modifizierte Installations DVD's haben eventuell einen anderen Aufbau und funktionieren nicht.
- Sollten Sie aus irgendwelchen Gründen das Verzeichnis `/var/lib/opsi/depot/opsi_nfs_share` nicht vom opsi-server aus per NFS exportieren können (z.B. weil der Depotshare vom opiserver per NFS von einem NAS eingebunden ist), so kann der zu verwendende NFS-share über ein Serverweites config angegeben werden. Z.B. `clientconfig.opsi_nfs_share=172.16.166.1:/var/lib/opsi/depot/opsi_nfs_share`

Die opsi v4.0.6 Netbootprodukte für Debian und Ubuntu beziehen Ihre Installations-Dateien nicht aus einem ISO-File. Vielmehr werden diese von uns mit dem Standard Netboot-Kernel und initrd ausgeliefert. Alle weiteren benötigten Pakete werden über das Internet bezogen. Zur Entlastung Ihrer Netzwekverbindung kann bei vielen Installationen daher die Verwendung eines lokalen apt-cache sinnvoll sein.

Die Pakete `debian8` und `ubuntu16-04` können auch auf ein lokales http-Repository zugreifen.

Siehe auch Kapitel Abschnitt [9.5.14](#)

Siehe auch Kapitel Abschnitt [9.5.6](#)

Startreihenfolge beteiligter Dienste unter SLES 11

Es kann vorkommen, dass der `showmount`-Befehl mit einer Fehlermeldung wie nachfolgend abbricht:

```
# showmount -e localhost
clnt_create: RPC: Program not registered
```

Bitte stellen Sie sicher, dass nach der Installation des NFS-Servers ein Neustart stattgefunden hat. Anschließend müssen die Dienste `rpcbind` und `nfsserver` in genau dieser Reihenfolge gestartet werden.

Ein Neustart der Dienste kann wie folgt durchgeführt werden:

```
# service rpcbind restart
# service nfsserver restart
```

Anschließend liefert `showmount` das gewünschte Ergebnis:

```
# showmount -e localhost
Export list for localhost:
/var/lib/opsi/depot/opsi_nfs_share *
```

Allgemeine Properties der opsi v4.0.6 Linux Netboot Produkte

Die folgenden Properties finden Sie zur Steuerung der Linuxinstallation in allen v406 Netbootprodukten:

- **askbeforeinst:**
Soll das Starten der Installation am Client bestätigt werden müssen? (Default=`true`)
- **architecture:**
Mit welcher Architektur soll das Zielsystem installiert werden?
Beeinflusst außerdem das verwendete Bootimage. (Default=`64bit`)
- **language oder locale:**
Welche Sprache / locale soll installiert werden. (Default=Distributionsabhängig / `de`)

- **console_keymap:**
Zu installierendes Tastaturlayout. (Default=Distributionsabhängig / *de*)
- **timezone:**
Welche Zeitzone soll verwendet werden?. (Default=*Europe/Berlin*)
- **root_password:**
Passwort für root. (Default=*linux123*)
- **user_password:**
Passwort für user. (Default=*linux123*)
- **proxy:**
Proxystring (wenn benötigt) in der Form: `http://<ip>:<port>`. (Default="")
- **install_opsi-client-agent:**
Installiere den opsi-client-agent für Linux (Kofinanzierungsprojekt: Sie benötigen eine Aktivierung durch die `/etc/opsi/modules`). (Default=*true*)
- **setup_after_install:**
Welche opsi-Produkte sollen zum Abschluss der Betriebssysteminstallation auf **setup** gestellt werden. (Default="")

Die Produkte: **debian7** , **debian8** und **ubuntu14-04**, **ubuntu16-04**

Die Basis-Installation erfolgt direkt aus dem Netz. Bei **debian8** und **ubuntu16-04** ist auch eine Installation von einem lokalen Repository möglich.

Das Produkt hat produktiven Status.

Das Produkt hat folgende zusätzliche Properties:

- **online_repository:**
Repository der Distribution für die Installation. (Nur bei Debian/Ubuntu Produkten)
(Default=Distributionsabhängig)
- **encrypt_password:**
Passwort für die Festplattenverschlüsselung (nur verwendet wenn `encrypt_logical_volumes=true`)
Example: `linux123` Default: `linux123`
- **partition_disk:**
Zu verwendende Festplatte: **first** oder kompletter device path Examples: "first", "/dev/sda", "/dev/sdb"
Default: `first`
- **partition_method:**
Methode zur Partitionierung der Festplatte:
regular: Standard Partionierung / **lvm**: LVM's anlegen / **crypto**: In einer verschlüsselten Partition LVM's anlegen
Possible: "regular", "lvm", "crypto"
Default: `lvm`
- **partition_recipe:**
Die Art der verwendeten Partitionierung:
atomic: Alles in einer Partition / **home**: eigene /home Partition / **multi**: eigene /home, /usr, /var, und /tmp Partitionen
Possible: "atomic", "home", "multi"
Default: `atomic`
- **desktop_package:**
Zu installierendes desktop package (standard = kein desktop) Possible: "standard", "ubuntu-desktop", "kubuntu-desktop", "lubuntu-desktop", "xubuntu-desktop", "ubuntu-gnome-desktop"
Default: `standard`

- **language_packs:**
Possible: "ar", "bg", "by", "cf", "de", "dk", "en", "es", "et", "fa", "fi", "fr", "gr", "il", "it", "kg", "kk", "lt", "mk", "nl", "no", "pl", "ro", "ru", "sg", "sr", "ua", "uk", "us", "wo"
Default: de

Videos (Zeitraffer) Folgende Videos zeigen jeweils eine Installation.

Sie sind mit einem Frame pro Sekunde aufgenommen und dadurch schneller anzusehen als die Installation eigentlich dauert.

- [Debian 7](#)
- [Debian 8](#)
- [Ubuntu 14.04](#)

Das Produkt ucs41 und ucs420

Die Basis-Installation bezieht ihre Pakete von den offiziellen UCS Repositories. Eine Installation mit lokalen Paketquellen ist ebenfalls möglich.

Dieses produkt hat einen produktiven Status.

Mit diesem Produkt ist es möglich einen Master-, Slave-, Backup, und einen Member-Server zu installieren. Wir empfehlen das l-opsi-server Produkt um aus einer UCS Maschine auch einen opsi-Server zu machen. Dieses Produkt ermöglicht es auch Clients über einen Member-Server zu installieren, hierfür werden einige Besonderheiten durchgeführt.

Das Produkt hat über die oben genannten Properties eines z.B debian8 Produktes noch die folgenden zusätzlichen UCS spezifischen Properties:

- **dns_domain:**
Der DNS Domain Name:
Example: `example.com` Default: `ucs.test`
- **ldap_base:**
ldap base. Example: `dc=example,dc=com` Default: `dc=ucs,dc=test`
- **ucs_code_name:**
Der Codename der UCS-Version welche im onlien Repository bereit gestellt wird.
Example: `ucs414` Default: `ucs414`
- **organisation:**
Der Name der Organisation der bei der UCS Installation verwendet wird.
Example: `uib gmbh` Default: `uib gmbh`
- **windomain:**
Der Name der Samba/Windows Domain.
Example: `MYDOMAIN` Default: `MYDOMAIN`
- **external_nameserver:**
Welcher externe Nameserver soll bei der Installation verwendet werden ?
Example: `10.11.12.13` Default: `auto` = the name server given by dhcp
- **ucs_master_ip:**
Die IP-Nummer des UCS Domain Controller (wird beim joinen von anderen Rollen verwendet) ?
Example: `10.10.10.10` Default: `10.10.10.10`
- **ucs_master_admin_password:**
Das Administrator Passwort des UCS Domain Controller (wird beim joinen von anderen Rollen verwendet) ?
Example: `linux123` Default: `linux123`

- **ucs_role:**
Welche UCS Rolle soll installiert werden ?
Possible: "domaincontroller_master", "domaincontroller_backup", "domaincontroller_slave", "memberserver", "base"
Default: domaincontroller_master

Einrichtung eines lokalen deb http Repository

Mit dem debian8, ubuntu16-04 und ucs41 Paket ist es nun möglich von einem lokalen Apache2 Repository zu installieren.

Dazu müssen bei dem Produkt im Property *online_repository* die entsprechende Adresse angeben nach dem Muster `http://<opsi-server>/opsi/<productId>` z.B `http://opsiserver/opsi/debian8`

Weiterhin muss das lokale Repository natürlich erstellt werden.

Stellen Sie dazu sicher, dass das Produkt `opsi-linux-support` auf Ihrem opsi-server installiert ist. Dieses Paket installiert die hierfür benötigten Distributions-Pakete (apache2) und erstellt auch die benötigten Ordner. Dieser muss danach mit einem passenden distributions Repository gefüllt werden.

Hierfür gibt es zwei Möglichkeiten:

1. Einfach: Sie laden sich ein von uns gebautes und getestetes Repository herunter und packen aus
2. Aufwendiger: Sie bauen es sich selbst.

Einfach:

Führen Sie das nachfolgende Script als *root* aus.

Beachten Sie das der Pfad zum Apache2 DocumentRoot zum einen Distributionstypisch unterschiedliche Defaults hat und darüberhinaus abweichend vom Default konfiguriert sein kann.

Daher müssen Sie evtl. die zweite Zeile des Scriptes anpassen !

debian8

```
#!/bin/bash
DOCUMENTROOT=/var/www/html
URL=http://download.uib.de/opsi4.0/products/opsi-linux
FILE=debian8.tgz
mkdir -p ${DOCUMENTROOT}/opsi
cd ${DOCUMENTROOT}/opsi
wget ${URL}/${FILE}
tar xzf ${FILE}
opsi-set-rights .
```

ubuntu16-04

```
#!/bin/bash
DOCUMENTROOT=/var/www/html
URL=http://download.uib.de/opsi4.0/products/opsi-linux
FILE=ubuntu16-04.tgz
mkdir -p ${DOCUMENTROOT}/opsi
cd ${DOCUMENTROOT}/opsi
wget ${URL}/${FILE}
tar xzf ${FILE}
opsi-set-rights .
```

ucs41

```
#!/bin/bash
DOCUMENTROOT=/var/www/html
URL=http://download.uib.de/opsi4.0/products/opsi-linux/univention-repository/
FILE=univention-repository-4.1.tgz
mkdir -p ${DOCUMENTROOT}/opsi
cd ${DOCUMENTROOT}/opsi
wget ${URL}/${FILE}
tar xzf ${FILE}
opsi-set-rights .
```

ucs42

```
#!/bin/bash
DOCUMENTROOT=/var/www/html
URL=http://download.uib.de/opsi4.0/products/opsi-linux/univention-repository/
FILE=univention-repository-4.2.tgz
mkdir -p ${DOCUMENTROOT}/opsi
cd ${DOCUMENTROOT}/opsi
wget ${URL}/${FILE}
tar xzf ${FILE}
opsi-set-rights .
```

Bitte beachten Sie die Datei:

<http://download.uib.de/opsi4.0/products/opsi-linux/univention-repository/opsi-ucs-repository-readme.txt>

Aufwendiger:

Sie können das Repository auch selbst erstellen:



Achtung

Ein selbst erstelltes Repo auf Basis einer UCS 4.2-0 DVD führt zu einem unvollständigem Repository. Hierbei ist das Paket debootstrap nicht fähig ein UCS 4.2-0 zu installieren. Das von uns bereitgestellte Repository ist hiervon nicht betroffen.

```
#!/bin/bash
set -x
BASE_DIR=/var/www/opsi
DVD_PATH=UCSISOMOUNTPOINT
UCS_VERSION=4.1
UCS_SUBVERSION=4
UCS_REPODIR=univention-repository/mirror
UCS_REPODIR2=${UCS_VERSION}/maintained/${UCS_VERSION}-${UCS_SUBVERSION}
UCS_RELEASE_PATH=dists/ucs414/main/binary-amd64/Release

cd ${BASE_DIR}
mkdir -p ${UCS_REPODIR}
cd ${UCS_REPODIR}
pwd
ln -s . univention-repository
mkdir -p ${UCS_REPODIR2}
cd ${UCS_REPODIR2}
pwd
cp -r ${DVD_PATH}/all .
cp -r ${DVD_PATH}/amd64 .
cp -r ${DVD_PATH}/dists .
mkdir -p i386
cd all
dpkg-scanpackages . /dev/null | gzip -9c > Packages.gz
dpkg-scanpackages . /dev/null > Packages.gz
cd ..
cd amd64
dpkg-scanpackages . /dev/null | gzip -9c > Packages.gz
```

```

dpkg-scanpackages . /dev/null > Packages.gz
cd ..
cd i386
dpkg-scanpackages . /dev/null | gzip -9c > Packages.gz
dpkg-scanpackages . /dev/null > Packages.gz
cd ..
echo "Archive: stable" > ${UCS_RELEASE_PATH}
echo "Origin: Univention" >> ${UCS_RELEASE_PATH}
echo "Label: Univention" >> ${UCS_RELEASE_PATH}
echo "Version: ${UCS_VERSION}.${UCS_SUBVERSION}" >> ${UCS_RELEASE_PATH}
echo "Component: main" >> ${UCS_RELEASE_PATH}
echo "Architecture: amd64" >> ${UCS_RELEASE_PATH}
cat ${UCS_RELEASE_PATH}
cd ${BASE_DIR}
chown -R www-data:www-data univention-repository
echo "all done"

```

Die Produkte sles11sp4, sles12, sles12sp1

Das Produkt hat folgende zusätzliche Properties:

```

name: productkey
multivalue: False
editable: True
description: email:regcode-sles for suse_register. Is only used if the
host parameter 'license-management.use' is set to false . If it set to True
the license key will be get from the license management module. / La clé de licence pour l
values: ["", "myemail@example.com:xxxxxxxxxxxxxxxx"]
default: [""]

name: suse_register
description: set to false , if you don't want to register your system online , if you set th
default: True

name: local_repositories
multivalue: True
editable: True
description: list of local repositories to use. Syntax: "repository description", example
values: [""]
default: [""]

name: install_unattended
description: If false then do interactive installation
default: True

```

Installationsquelle Zum herunterladen der Installations DVD brauchen Sie einen Account bei SUSE. Installations DVD sollte heissen (mit einer Datei dieses Namens haben wir getestet): sles11sp4: SLES-11-SP4-DVD-x86_64-GM-DVD1.iso sles12: SLE-12-Server-DVD-x86_64-GM-DVD1.iso sles12sp1: SLE-12-SP1-Server-DVD-x86_64-GM-DVD1.iso ISO-File kopieren nach /var/lib/opsi/depot/opsi_nfs_share/opensusel42-1/64/ Ausführung von opsi-set-rights nicht vergessen.

Videos (Zeitraffer) Folgendes Video zeigt eine Installation.

Es ist mit einem Frame pro Sekunde aufgenommen und dadurch schneller anzusehen als die Installation eigentlich dauert.

- [Suse Linux Enterprise Server 12](#)

Die Produkte redhat70 und centos70

Das Produkt hat folgende zusätzliche Properties:

```

name: install_unattended
description: If false then do interactive installation
default: True

name: selinux_mode
multivalue: False
editable: False
description: In which mode should SELinux run ?
values: ["enforcing", "permissive", "disabled"]
default: ["permissive"]

name: partition_method
multivalue: False
editable: False
description: plain: Regular partitions with no LVM or Btrfs. / lvm: The LVM partitioning s
values: ["plain", "lvm", "btrfs", "thinp"]
default: ["lvm"]

name: productkey
multivalue: False
editable: True
description: email:regcode for subscription_register. Is only used if the
host parameter 'license-management.use' is set to false. If it set to True
the license key will be get from the license management module. / La clé de licence pour l
values: ["", "myemail@example.com:xxxxxxxxxxxxxxxx"]
default: [""]

name: subscription_register
description: set to false, if you don't want to register your system online, if you set th
default: True

```

Installationsquelle CentOS Installations DVD hier herunterladen: z.B.:

http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-DVD-1511.iso ISO-File kopieren nach /var/lib/opsi/depot/opsi_nfs_share/centos70/64/ Ausführung von opsi-set-rights nicht vergessen.

Installationsquelle RedHat Zum herunterladen der Installations DVD brauchen Sie einen Account bei RedHat. Installations DVD sollte heißen (mit einer Datei dieses Namens haben wir getestet):

rhel-server-7.0-x86_64-dvd.iso ISO-File kopieren nach /var/lib/opsi/depot/opsi_nfs_share/redhat70/64/ Ausführung von opsi-set-rights nicht vergessen.

Videos (Zeitraffer) Folgende Videos zeigen eine Installation.

Sie sind mit einem Frame pro Sekunde aufgenommen und dadurch schneller anzusehen als die Installation eigentlich dauert.

- [CentOS 7](#)
- [RedHat 7](#)

9.5.7 Linux Netboot Produkte v4.0.5 ohne Distributionseigenen Installer

Basis-Installation des OS per Netboot

Für die Installation eines Linux Basissystems wird zunächst per Netboot das Standard opsi-linux-bootimage gebootet (welches auch für die Windows-Installationen zum Einsatz kommt).

Von diesem Bootimage aus wird die Ziel-Festplatte partitioniert (/ und swap) und formatiert. Nun folgt die Installation des Grundsystems (mit Netzwerkkonfiguration und ssh aber ohne X11). Die Abläufe dieser Grundinstallation unterscheiden sich naturgemäß zwischen den unterschiedlichen Distributionen erheblich. Gemeinsam ist, dass die Installation direkt aus den Originalpaketen der Distribution erfolgt.

Auf diese Basisinstallation können optional die opsi-Pakete installiert werden, um aus dem System einen opsi-Server (z.B. neuen Depotservers) zu machen.

Ebenfalls optional kann nun der opsi-client-agent für Linux installiert werden. Dieser ist dann für die Installation und Konfiguration weiterer Software zuständig.

Die opsi-Netboot-Produkte zur Linuxinstallation sind bereits als Open Source freigegeben.

Bedingt dadurch, dass die Basisinstallation aus dem Standard opsi-linux-bootimage erfolgt, gibt es distributionsabhängig unterschiedlich bestimmte Dinge, welche sich erst in der Umgebung nach dem ersten Boot des Systems konfigurieren bzw. installieren lassen. Beispiele hierfür sind die SELinux-Installation bei den *RedHat artigen* bzw. die Konfiguration der Tastatur bei den *Debian artigen*. Hierfür gibt es ein Standard Localbootprodukt `l-os-postinst` welches diese Aufgaben übernimmt.

Allgemeine Properties der v4.0.5 Linux Netboot Produkte

Die folgenden Properties finden Sie zur Steuerung der Linuxinstallation in allen Netbootprodukten:

- **askbeforeinst:**
Soll das Starten der Installation am Client bestätigt werden müssen? (Default=*true*)
- **architecture:**
Mit welcher Architektur soll das Zielsystem installiert werden?
Beeinflusst die Auswahl des bootimages und die Installationsarchitektur. (Default=*64bit*)
- **system_partition_size:**
Größe der Systempartition. Die Größe kann in Prozent der Festplattengröße oder als absoluter Wert (G=Gigabyte) angegeben werden. Wenn Sie einen kleineren Wert als 100% angeben, wird der verbleibende Rest als Datenpartition verwendet (wenn das Property `data_partition_create = true`). (Default=*100%*)
- **swap_partition_size:**
Größe der Swappartition. (Default=*2000M*)
- **data_partition_create:**
Verwende freien Plattenplatz zur Erstellung einer Datenpartition. (true/false). (Default=*true*)
- **data_partition_preserve:**
Soll eine existierende Datenpartition erhalten werden ?
always = Installation abbrechen wenn der Erhalt einer gefundenen Partition mit dem Label *data* mit den angegebenen Partitionierungsdaten nicht möglich ist.
if_possible = Wird eine Partition mit dem Label *data* gefunden und der Erhalt dieser Partition ist gemäß der angegebenen Partitionierungsdaten nicht möglich so wird die Partition gelöscht.
never = Die gesamte Partitionstabelle wird immer neu geschrieben. (Default=*never*)
- **language:**
Welche Sprache / locale soll installiert werden. (Default=*de*)
- **console_keymap:**
Zu installierendes Tastaturlayout. (Default=Distributionsabhängig / *de*)
- **timezone:**
Welche Zeitzone soll verwendet werden?. (Default=*Europe/Berlin*)
- **root_password:**
Passwort für root. (Default=*linux123*)
- **user_password:**
Passwort für user. (Default=*linux123*)

- **install_opsi_server:**
Installiere die opsi-server Pakete. (Default=*false*)
- **online_repository:**
Repository der Distribution für die Installation. (Nicht bei SLES) (Default=Distributionsabhängig)
- **opsi_online_repository:**
Repository der opsi-server Pakete. (Default=Distributionsabhängig)
- **proxy:**
Proxystring (wenn benötigt) in der Form: `http://<ip>:<port>`. (Default="")
- **additional_packages:**
Welche zusätzlichen Pakete sollen installiert werden? Angabe der Pakete Leerzeichen separiert. (Default="")
- **wget_and_execute:**
Url (http) einer Datei welche am Ende der Installation geholt und ausgeführt wird. (Default="")
- **install_opsi-client-agent:**
Installiere den Linux opsi-client-agent (Kofinanzierungsprojekt: Sie benötigen eine Aktivierung durch die `/etc/opsi/modules`) . (Default=*false*)
- **release:**
(nur Debian und Ubuntu)
Welches Release der Distribution soll installiert werden. (Default=Distributionsabhängig)
- **setup_after_install:**
Welche opsi Produkte sollen zum Abschluss der Betriebssysteminstallation auf setup gestellt werden. (Default=*l-os-postinst*)

ubuntu

Die Basis Installation erfolgt per debootstrap direkt aus dem Netz.

Das Produkt hat produktiven Status.

Das Produkt ist UEFI/GPT kompatibel (getestet für `release=trusty`).

Es gibt für diese Produkt passende opsi-server Pakete welche über `install_opsi_server=true` installiert werden können.

debian

Die Basis Installation erfolgt per debootstrap direkt aus dem Netz.

Das Produkt hat produktiven Status.

Das Produkt ist UEFI/GPT kompatibel (getestet für `release=wheezy`).

Es gibt für diese Produkt passende opsi-server Pakete welche über `install_opsi_server=true` installiert werden können.

9.5.8 opsi-linux-client-agent

Der opsi-client-agent für Linux ist Bestandteil des Kofinanzierungsprojektes *Linux Agent* und derzeit kostenpflichtig.



Achtung

Dies ist ein Problem mit der derzeitigen Datenstruktur in opsi und wird zu einem späteren Zeitpunkt gelöst.

Der opsi-client-agent für Windows besteht im Kern aus den Komponenten:

1. dem Service `opsiclientd`
2. dem Hilfsprogramm `opsiscriptstarter`
3. dem Actionprocessor `opsi-script` / `opsi-script-nogui`

Der opsi-client-agent für Linux basiert auf einer Portierung des Windows-Clientagenten nach Linux.

Der `opsiclientd` ist derzeit nicht für alle unterstützten Distributionen verfügbar und ist deshalb für alle anderen Distributionen zunächst durch das Programm `opsiscriptstarter` ersetzt

Der `opsiclientd` steht auf folgenden Distributionen / Releases zur Verfügung:

- Debian 7 / 8
- Ubuntu 12.04 / 14.04 / 16.04
- openSuse 13.2 / 42.1
- SLES 12 / 12SP1
- UCS 4.0 / 4.1

Steht kein `opsiclientd` zur Verfügung, wird der `opsiscriptstarter` so installiert, das er die Aufgaben des `opsiclientd` beim Systemstart übernimmt:

- Kontakt mit dem opsi-Server: Prüfen ob Aktionen gesetzt sind
- Mounten des Depot Shares
- Starten des Actionprocessors
- Unmount des Depot Shares
- Senden der Logdatei an den Server

Der Actionprocessor heißt unter Linux `opsi-script` und ist aus den selben Quellen gebaut wie der `opsi-winst` unter Windows. Damit steht unter Linux die gleiche Scriptsyntax zur Verfügung wie unter Windows. Weiterhin sind alle nicht plattformspezifischen Funktionen umgesetzt wie z.B:

- File handling
- String und Stringlisten Funktionen
- Ausführen von externen Scripten und Programmen
- Kommunikation mit dem opsi-Server
- Patchen von Konfigurationsdateien

Natürlich gibt es unter Linux keine Funktionen zum Patchen der Registry, dafür aber neue linuxspezifische Funktionen wie z.B.:

- `getLinuxDistroType`
- `getLinuxVersionMap`

Das Logging des `opsi-script` ist analog zur dem des `opsi-winst` unter Windows.

Anders als bei Windows gibt es den `opsi-script` neben einer grafischen Version für die Arbeit unter X-Windows zusätzlich in einer Version *no-gui* für Systeme ohne grafische Oberfläche

opsi-linux-client-agent: Installation: service_setup.sh

Diese Methode dient zur Installation auf einzelnen Rechnern. Für ein Massen-Rollout siehe weiter unten.

1. Loggen Sie sich mit `root` Rechten auf dem Client ein.
2. Mounten Sie den share `//<opsiserver>/opsi_depot` an eine beliebige Stelle.
3. Wechseln Sie in das Verzeichnis `opsi-linux-client-agent` auf dem gemounteten share
4. Starten Sie dort das Script `./service_setup.sh`



Achtung

Der Client rebootet nach der Installation.

opsi-linux-client-agent: Installation: opsi-deploy-client-agent

Das `opsi-deploy-client-agent` Skript verteilt den `opsi-client-agent` direkt vom `opsi-server` auf die Clients.

Voraussetzung auf dem `opsi-server`:

- Das Python 2-Paket **paramiko** muss installiert sein. Dieses steht für die meisten Distributionen als `python-paramiko` zur Verfügung und kann über den jeweiligen Paketmanager installiert werden.

Voraussetzung hierfür sind bei den Clients:

- `ssh` Zugang als `root` oder als `user` der `sudo` ohne Passwort-Eingabe ausführen darf

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

Mit dem `opsi-deploy-client-agent` Skript kann auch eine ganze Liste von Clients bearbeitet werden. Dazu können entweder beliebig viele Clients als letzter Parameter übergeben werden oder mit der Option `-f` die Clients aus einer Datei eingelesen werden. Bei der Verwendung einer Datei, muss in jeder Zeile ein Client stehen.

Das Script kann mit IP-Adressen, Hostnamen und FQDNs arbeiten. Es wird versuchen automatisch zu erkennen welche Art von Adresse übergeben wurde.

Mit dem `opsi-deploy-client-agent` Skript kann auch eine ganze List von Clients bearbeitet werden. Das Skript findet sich unter `/var/lib/opsi/depot/opsi-linux-client-agent`

Führen Sie das Script mit `root` Rechten aus.

```
bonifax:/var/lib/opsi/depot/opsi-linux-client-agent# ./opsi-deploy-client-agent --help
usage: opsi-deploy-client-agent [-h] [--version] [--verbose]
                                [--debug-file DEBUGFILE] [--username USERNAME]
                                [--password PASSWORD]
                                [--use-fqdn | --use-hostname | --use-ip-address]
                                [--ignore-failed-ping]
                                [--reboot | --shutdown | --start-opsiclientd]
                                [--hosts-from-file HOSTFILE]
                                [--skip-existing-clients]
                                [--threads MAXTHREADS]
                                [--keep-client-on-failure | --remove-client-on-failure]
                                [host [host ...]]
```

Deploy opsi client agent to the specified clients. The clients must be accessible via SSH. The user must be allowed to use `sudo` non-interactive.

positional arguments:

```

host                The hosts to deploy the opsi-client-agent to.

optional arguments:
-h, --help          show this help message and exit
--version, -V       show program's version number and exit
--verbose, -v       increase verbosity (can be used multiple times)
--debug-file DEBUGFILE
                    Write debug output to given file.
--username USERNAME, -u USERNAME
                    username for authentication (default: root). Example
                    for a domain account: -u "<DOMAIN>\\<username>"
--password PASSWORD, -p PASSWORD
                    password for authentication
--use-fqdn, -c      Use FQDN to connect to client.
--use-hostname      Use hostname to connect to client.
--use-ip-address    Use IP address to connect to client.
--ignore-failed-ping, -x
                    try installation even if ping fails
--reboot, -r        reboot computer after installation
--shutdown, -s     shutdown computer after installation
--start-opsiclientd, -o
                    start opsiclientd service after installation
--hosts-from-file HOSTFILE, -f HOSTFILE
                    File containing list of clients (one hostname per
                    line). If there is a space followed by text after the
                    hostname this will be used as client description for
                    new clients.
--skip-existing-clients, -S
                    skip known opsi clients
--threads MAXTHREADS, -t MAXTHREADS
                    number of concurrent deployment threads
--keep-client-on-failure
                    If the client was created in opsi through this script
                    it will not be removed in case of failure. (DEFAULT)
--remove-client-on-failure
                    If the client was created in opsi through this script
                    it will be removed in case of failure.

```

opsi-linux-client-agent: Installation: Durch die opsi netbootprodukte

Wenn Sie ein Linux über die opsi-Netboot-Produkte installieren, wird der opsi-linux-client-agent automatisch mit installiert, wenn das Property `install_opsi-client-agent` auf `true` steht.

opsi-linux-client-agent: opsiclientd Konfiguration

Der `opsiclientd` für Linux ist eine Portierung des `opsiclientd` für Windows und arbeitet mit einer analogen Konfigurations Datei: `/etc/opsi-client-agent/opsiclientd.conf`.

Eine ausführliche Beschreibung dieser Konfiguration findet sich im Kapitel zum opsi-client-agent: Abschnitt [6.1.3](#)

Dabei sind nicht alle Features und Events auch unter Linux verfügbar.

Verfügbar sind:

- Start beim Systemstart (bzw. start des `opsiclientd`) unter Linux ist der Name des Events `opsiclientd_start` (und nicht `gui_startup`)
- `event_on_demand`
- Das `event_timer` aber nur mit der Einstellung: `super = default`

Nicht verfügbar sind (derzeit):

- Alles was mit dem lokalen Cache (*WAN-Erweiterung*) zu tun hat.

- Die Modifikation der Events durch Preconditions
- Der opsiclientd Notifier
- Das event_net_connection
- Das event_on_shutdown
- Das event_silent_install

opsi-linux-client-agent: Pfade

Der opsi-linux-client-agent legt Dateien an folgenden Orten ab:

Die Binaries:

/usr/bin/opsi-script (X11)

/usr/bin/opsi-script-nogui (ohne X11)

/usr/bin/opsiscriptstarter (Hilfsprogramm bzw. opsiclientd Ersatz)

/usr/bin/opsiclientd

Die Hilfsdateien für den opsi-script finden sich in:

Skindateien:

/usr/share/opsi-client-agent/opsi-script/skin (depricated)

Ab opsi-script 4.12.0.31 / opsi-linux-client-agent (4.1.0.9-1):

default : /usr/share/opsi-script/skin

custom : /usr/share/opsi-script/customskin

opsi-script Library:

/usr/share/opsi-script/lib

Translation Dateien:

/usr/share/locale/<LANG>/LC_MESSAGES/opsi-script.po

Die Konfigurationen:

/etc/opsi-client-agent/opsiclientd.conf (Konfiguration des opsiscriptstarter / opsiclientd)

/etc/opsi-client-agent/opsi-script.conf (depricated)

Ab opsi-script 4.12.0.31 / opsi-linux-client-agent (4.1.0.9-1):

/etc/opsi-script/opsi-script.conf

Logdateien sind zu finden unter:

/var/log/opsi-client-agent

/var/log/opsi-client-agent/opsiclientd

/var/log/opsi-client-agent/opsi-script (depricated)

Ab opsi-script 4.12.0.31 / opsi-linux-client-agent (4.1.0.9-1):

/var/log/opsi-script/

opsi-linux-client-agent: Known Bugs

Das Kopieren von vielen Dateien von einem Samba3-Share ist abhängig von der Samba-Version fehlerhaft. Es werden nicht alle Dateien kopiert. Das Problem wurde bei Samba4 Shares bisher nicht beobachtet.

Als Workaround kann statt:

```
[Files_copy_netboot]
copy -s "%scriptPath%/installfiles/*" "$target$/installfiles/"
```

das folgende verwendet werden:

```
[ShellInAnIcon_opsi_copy_netboot]
set -x
export PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin
cd "%scriptPath%"
tar cf - installfiles | ( cd "$target$/installfiles/" ; tar xf - )
```

9.5.9 Beispiel Scriptteile

Unter Windows gilt für die Softwareverteilung: Die Installation von Software ist genauso wichtig wie die anschließende Konfiguration der Software.

Unter Linux stehen die meisten Pakete über die Repositories der Distribution zur Verfügung. Dadurch wird der Installationsanteil kleiner, der Konfigurationsanteil aber bleibt. Weiterhin gibt es auch Applikationen, welche nicht über die Standardrepositories verfügbar sind.

Hier müssen unter Umständen zunächst weitere Repositories dem System hinzugefügt werden bzw. Installationsquellen dem Paket zugefügt werden.

Wichtig ist, dass alle Installations- und Konfigurationsarbeiten zentral vom opsi-Server gesteuert und dort auch geloggt werden.

Im folgenden finden Sie Beispiele für folgende Aufgaben in einem beispielhaften Script für den opsi-linux-client-agent:

- Beenden wenn es nicht unter Linux läuft
- Feststellen des Distributionstyps zur Entscheidung zwischen `apt-get`, `zypper` und `yum`
- Feststellen der genauen Linux Version
- Installation eines Paketes
- Hinzufügen eines Repositories

Beispiel: Beenden wenn es nicht unter Linux läuft:

```
[Actions]
requiredWinstVersion >= "4.11.4.1"
ScriptErrorMessages=off

DefVar $OS$

set $OS$ = GetOS

if not($OS$ = "Linux")
    LogError "Wrong OS: Product: " + $ProductId$ + " is only for Linux"
    isFatalError "Wrong OS"
endif
```

Beispiel: Feststellen des Distributionstyps:

```
[Actions]
requiredWinstVersion >= "4.11.4.1"
ScriptErrorMessages=off

DefVar $distrotype$

set $distrotype$ = getLinuxDistroType

if $distrotype$ = 'debian'
```

```

    Message "Try to get Package Lock..."
    if waitForPackageLock("60","false")
        comment "we got the package lock."
    else
        LogError "could not get Package Lock"
        isFatalError "package lock failed"
    endif
    ShellInAnIcon_Upgrade_deb
else
    LogError "Wrong Distro: This Product is for Debian/Ubuntu only"
    isFatalError "Wrong distro"
endif

if not("0" = getLastExitCode)
    Message "failed ShellInAnIcon_Upgrade"
    LogError "failed ShellInAnIcon_Upgrade"
    isFatalError "failed Upgrade"
endif

[ShellInAnIcon_Upgrade_deb]
set -x
export DEBIAN_FRONTEND=noninteractive
apt-get --yes install aptitude
apt-get update
apt-get --yes dist-upgrade
exit $?

```

Beispiel: Feststellen der genauen Linux Version und Installation eines Paketes:

```

[Actions]
requiredWinstVersion >= "4.11.4.1"
ScriptErrorMessages=off

DefVar $distCodeName$
DefVar $distroName$
DefVar $distRelease$
DefVar $desktop$

DefStringList $linuxInfo$

set $linuxInfo$ = getLinuxVersionMap
set $distCodeName$ = getValue("Codename", $linuxInfo$)
set $distRelease$ = getValue("Release", $linuxInfo$)
set $distroName$ = getValue("Distributor ID", $linuxInfo$)

set $desktop$ = GetProductProperty("desktop", "kde")

if $distrotype$ = 'suse'
    if $desktop$ = "unity"
        Message " No Unity on SUSE - fallback to KDE ..."
        set $desktop$ = "kde"
    endif ; unity

    Message "Try to get Package Lock..."
    if waitForPackageLock("60","false")

```

```

        comment "we got the package lock."
    else
        LogError "could not get Package Lock"
        isFatalError "package lock failed"
    endif

    if $desktop$ = "kde"
        if ($distroName$ = 'openSUSE project')
            ShellInAnIcon_kde_suse
        endif
        if ("SUSE LINUX" = $distroName$) and ($distRelease$ = "11")
            ShellInAnIcon_kde_sles11
        endif
        if not("0" = getLastExitCode)
            LogError "failed ShellInAnIcon"
            Message "failed kde"
            isFatalError "failed kde"
        endif
    endif ; kde
endif; suse type

[ShellInAnIcon_kde_suse]
set -x
zypper --no-gpg-checks --non-interactive install patterns-openSUSE-kde4 patterns-openSUSE-
    kde4_basis
zypper --no-gpg-checks --non-interactive install splashy-branding-openSUSE
exit $?

[ShellInAnIcon_kde_sles11]
set -x
zypper --no-gpg-checks --non-interactive install --auto-agree-with-licenses -t pattern kde
exit $?

```

Beispiel: Hinzufügen eines Repositories:

```

[Actions]
requiredWinstVersion >= "4.11.4.1"
ScriptErrorMessage=off

DefVar $distCodeName$
DefVar $distroName$
DefVar $distRelease$
DefVar $desktop$

DefStringList $linuxInfo$

set $linuxInfo$ = getLinuxVersionMap
set $distCodeName$ = getValue("Codename", $linuxInfo$)
set $distRelease$ = getValue("Release", $linuxInfo$)
set $distroName$ = getValue("Distributor ID", $linuxInfo$)

set $desktop$ = GetProductProperty("desktop", "kde")

if $distroName$ = 'Ubuntu'

    if $desktop$ = "cinnamon"

```

```

set $desktopPackage$ = $desktop$
Message "Try to get Package Lock..."
if waitForPackageLock("60","false")
    comment "we got the package lock."
else
    LogError "could not get Package Lock"
    isFatalError "package lock failed"
endif
ShellInAnIcon_ubuntu_cinnamon
if not("0" = getLastExitCode)
    Message "failed ShellInAnIcon_ubuntu_cinnamon"
    LogError "failed ShellInAnIcon_ubuntu_cinnamon"
    isFatalError "failed cinnamon"
endif
endif ; cinnamon
endif; ubuntu

[ShellInAnIcon_ubuntu_cinnamon]
set -x
export DEBIAN_FRONTEND=noninteractive
# we need to get the add-apt-repository command
apt-get --yes --force-yes install python-software-properties
# the cinnamon repository
add-apt-repository ppa:gwendal-lebihan-dev/cinnamon-stable
apt-get update
apt-get --yes install ubuntu-desktop
exit $?

```

9.5.10 Linux Localboot Produkte

Hier einige Lokalbootprodukte welche zum Standardumfang des opsi Linuxsupports gehören.

Das Produkt l-opsi-server

Das Produkt *l-opsi-server* dient dazu automatisiert auf einer Linuxmaschine per opsi-linux-client-agent einen opsi-server zu installieren. Dies kann dazu dienen um schnell einen neuen opsi-depot-server zu installieren oder z.B. ein opsi Testsystem.

Achtung

Derzeit kann eine Maschine nicht gleichzeitig am selben opsi-config-server opsi-client und opsi-depot-server sein. Sie haben derzeit zwei Möglichkeiten mit dieser Beschränkung umzugehen:



1. Verwendung von einem opsi-config-server: Wenn also ein per *l-opsi-server* installierter opsi-server zum Depot-server an seinem Config-Server werden soll, so müssen Sie vorher im configed die Maschine als Client löschen.
2. Verwendung von zwei opsi-config-servern: Sie setzen für die Verwaltung Ihrer opsi-server einen zweiten, unabhängigen opsi-config-server auf, welcher nur dazu dient die anderen opsi-server zu installieren und zu pflegen. Dieser zweite opsi-config-server kennt die anderen opsi-server also nur als Clients, während der bisherige (erste) opsi-config-server die anderen opsi-server nur als Depots (oder garnicht) kennt.

In UCS Umgebungen wird Methode 2 empfohlen und der zweite opsi-config-server darf keine UCS Maschine sein.

Das Produkt *l-opsi-server* hat folgende Properties:

- opsi_online_repository:
(Basis-) Repository für die opsi-server installation.
(Default="http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40")
siehe auch *repo_kind*

- **opsi_noproxy_online_repository:**
 (Basis-) Repository für die opsi-server installation (ohne cache proxy).
 (Default="http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40")
 Sollten Sie bei **opsi_online_repository** einen Proxy oder deb-cacher mit angegeben haben (z.B. 'http://mydeb-cacher:9999/download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40'), dann geben Sie hier die URL nochmal ohne den Proxy an. Ansonsten geben Sie hier das selbe an wie bei **opsi_noproxy_online_repository**.
- **repo_kind:**
 Welche Repository Art ["experimental", "stable", "testing"] soll zur Installation verwendet werden ?. (Default=*stable*)
 Aus dem Client OS, *opsi_online_repository* und *repo_kind* wird die URL zusammengebaut welche verwendet um dem Client ein opsi Repository hinzuzufügen.
- **backend:**
 Welches Backend soll installiert werden ? (Die Auswahl **mysql** benötigt die Hinterlegung einer gültigen Freischalt-datei). (Default=*file*)
 Eine modules Datei mit den benötigten Freischaltungen kann im custom Verzeichnis des Produktes abgelegt werden. Wird dort eine modules gefunden so wird diese verwendet.
- **opsi_admin_user_name:**
 Unter welchen Namen soll ein opsi_admin_user erzeugt werden (empty= kein user wird erzeugt). (Default=*adminuser*)
 Wird hier ein user angegeben, so wird dieser angelegt, wird Mitglied der Gruppen *opsiadmin*, *pcpatch/opsifileadmin* und bekommt als unix- und samba Passwort den Wert von **opsi_admin_user_password**
- **opsi_admin_user_password:**
 Was ist das Passwort für den opsi_admin_user (empty= nicht erlaubt). (Default=*linux123*)
 siehe **opsi_admin_user_name**
- **setup_after_install:**
 Welche opsi Produkte sollen nach der Installation von l-opsi-server installiert werden ?. (Default="")
- **allow_reboot:**
 Darf die Maschine nach der Installation von l-opsi-server rebootet werden ?. (Default=*true*)
- **myipname:**
 Soll ein abweichender IP-Name (FQDN) bei der Installation verwendet werden ? (*auto*= use standard)
 (Default=*auto*)
 siehe **install_and_configure_dhcp**
- **myipnumber:**
 Soll eine abweichende IP-Nummer bei der Installation verwendet werden ? (*auto*= use standard) (Default=*auto*)+
 siehe **install_and_configure_dhcp**
- **install_and_configure_dhcp:**
 Soll ein DHCP-Server auf der Maschine installiert und konfiguriert werden ?. (Default=*False*)
 Wenn dieses Property *false* ist, so werden die Properties: *netmask*, *network*, *dnsdomain*, *nameserver* und *gateway* nicht beachtet, da diese nur der DHP Konfiguration dienen.
- **netmask:**
 Netmask (für dhcp). (Default="255.255.0.0")
 Nicht verwendet wenn *install_and_configure_dhcp=false*
- **network:**
 Netzwerk Adresse (für dhcp). (Default="192.168.0.0")
 Nicht verwendet wenn *install_and_configure_dhcp=false*
- **dnsdomain:**
 DNS domain (for dhcp). (Default="uib.local")
 Nicht verwendet wenn *install_and_configure_dhcp=false*

- **nameserver:**
Primary nameserver (für dhcp). (Default="192.168.1.245")
Nicht verwendet wenn *install_and_configure_dhcp=false*
- **gateway:**
gateway (option routers for dhcp). (Default="192.168.1.245")
Nicht verwendet wenn *install_and_configure_dhcp=false*
- **ucs_master_admin_password:**
Passwort des users *Administrators* auf dem UCS-Master.
Wird nur für UCS-Server benötigt und hier nur für alle Rollen ausser *Master*. (Default=*linux123*)
- **update_test:**
Nicht verwenden: Internal Debuging. (Default=*False*)

Das Produkt hat eine *setup required before* Abhängigkeit zu dem Produkt *l-system-update*. D.h. wenn Sie *l-opsi-server* auf *setup* stellen wird automatisch *l-system-update* auch auf *setup* gestellt und vorher installiert.

In dem Verzeichnis *custom* des Produktes *l-opsi-server* kann eine Freischaltdatei (*modules*) abgelegt werden, welche bei der Installation durch das Produkt *l-opsi-server* verwendet wird und beim Einspielen einer neuen Version des Produktes erhalten bleibt.

l-os-postinst für v4.0.5 Netboot installationen

Dieses Produkt übernimmt jene Teile der Basisinstallation welche sich vom Bootimage nicht korrekt ausführen lassen.

Dies ist für die unterschiedlichen Distributionen:

- CentOS:
 - Installation von SELinux

Das Produkt hat eine Abhängigkeit zu dem Produkt *l-system-update* welches vor dem Lauf von *l-os-postinst* aufgerufen wird.

Das Produkt hat eine hohe Priorität, d.h. es wird vor *normalen* Produkten ausgeführt.

l-desktop

Das Produkt *l-desktop* installiert einen Desktop auf dem Rechner.

Über das Property *desktop* kann der zu installierende Desktop ausgewählt werden. Dabei ist zu beachten, das nicht alle Desktops auf allen Distributionen verfügbar sind. So gibt es z.B. *Unity* nur unter Ubuntu. Wird ein nicht verfügbarer Desktop gewählt so wird ein Distributionsspezifischer Defaultdesktop installiert. Weiterhin haben die Desktop Pakete einen unterschiedlichen Umfang welcher abhängig von Distribution und Desktop sich auf die eigentliche Desktop Software beschränken kann oder auch Basisprodukte wie *libreoffice*, *firefox*, *PDF-Reader* usw. enthalten kann.

Das Property *desktop* hat folgende Werte:

- Gnome
Default für Debian, CentOS, RHEL
Verfügbar auf allen Distributionen.
- KDE
Default für SLES, OpenSuse Verfügbar auf allen Distributionen.
- Unity
verfügbar nur für Ubuntu
- Cinnamon
verfügbar nur für Ubuntu

- `xfce4`
Verfügbar auf Ubuntu, Debian.
- `lxde`
Verfügbar auf Ubuntu, Debian.

l-system-update

Dieses Produkt aktualisiert das System.

l-swaudit

Softwareinventarisierung auf Basis des Paketmanagers

l-hwaudit

Hardwareinventarisierung.

Die Hardwareinventarisierung basiert zur Zeit auf der in Python implementierten Methode wie sie auch vom bootimage verwendet wird. Dazu muß das Paket `python-opsi` aus dem opsi-Repository der Distribution installiert werden. Ist für die Distribution kein opsi-Repository verfügbar, so scheitert auch die Hardwareinventarisierung.

l-jedit

Java-basierter Editor mit Syntaxhighlighting für opsi-script. Ist noch kein Java installiert, so wird dieses automatisch installiert.

9.5.11 Inventarisierung

Zur Inventarisierung werden die Daten durch den Clientagenten erhoben und an den Server gesendet. Die Hardwareinventarisierung basiert auf den schon im Bootimage implementierten Methoden.

Die Softwareinventarisierung basiert auf den Daten des Paketmanagements der verwendeten Distribution.

9.5.12 UEFI / GPT Unterstützung

Einige der opsi v4.0.5 Linuxprodukte sind UEFI/GPT kompatibel. Details siehe hierzu in der obigen Auflistung der Netbootprodukte.

Die opsi v4.0.6 Linuxprodukte sind leider noch nicht UEFI/GPT kompatibel. Die Informationen über die UEFI Umgebung gehen im Moment noch bei dem kexec Boot verloren. Wir hoffen, das wir dieses Problem mit einem neueren bootimage irgendwann beheben können.

9.5.13 Roadmap

Die Linux Unterstützung von opsi ist neu. Das bedeutet auch, dass wir im ersten Release noch nicht alle Features verwirklicht haben.

Weitere Features werden folgen wie:

- Frei konfigurierbare Partitionierung
- Logical Volume Management
- Patchen von XML-Dateien
- Patchen von hierarchischen Konfigurationsdateien wie `dhcpd.conf`

9.5.14 Proxy für *.deb*-Pakete einrichten und verwenden

Anleitungen zur Erstellung eines eigenen Proxy zum Zwischenspeichern von *.deb*-Paketen finden Sie unter anderem hier:

- [Ubuntusers Wiki: Apt-Cacher-NG](#)
- [Gambaru.de: Apt-Cacher-NG: Ein Proxy-Server für Debian und Ubuntu](#)

9.6 opsi mit UEFI / GPT

9.6.1 Netboot-Produkte mit uefi Unterstützung:

(Stand / as of 25.08.2018)

Tabelle 9: opsi-clonezilla

Netboot product	Opsi 4.0.7 / 4.1	Remark
opsi-clonezilla	✓	

Tabelle 10: Standard Windows

Netboot product	Opsi 4.0.7 / 4.1	Remark
Server 2016	✓	
win10 64 Bit	✓	
win10 32 Bit	🚧	
Server 2012 R12	✓	
win8.1 64 Bit	✓	
win8.1 32 Bit	✗	
Server 2012	✓	
win7 64 Bit	✓	
win7 32 Bit	✗	
Server 2008 R2	✓	
winvista 32 Bit	✗	
winvista 64 Bit	⚠	
Server 2008 64 Bit	⚠	
winxp	✗	

✓ : Supported ✗ : Unsupported 🚧 : Under Development ⚠ : Discontinued

Tabelle 11: Linux

Tabelle 11: (continued)

Netboot product	Opsi 4.0.7 / 4.1	Remark
ubuntu	✓	
ubuntu18-04	✓	
ubuntu16-04	✓	since ubuntu16-04_4.0.7.2-1
ubuntu14-04	✗	
debian	✓	
debian9	✓	
debian8	✓	since debian8_4.0.7.2-1
debian7	🚧	
centos70	🚧	
centos65	⚠	
redhat70	🚧	
opensusel15	🚧	
opensusel42-3	✓	
opensusel42-2	⚠	
opensusel42-1	⚠	
opensuse13-2	⚠	
opensuse13-1	⚠	
sles15	🚧	
sles12sp1	🚧	
sles12	🚧	
sles11sp4	🚧	
sles11sp3	⚠	

✓ : Supported ✗ : Unsupported 🚧 : Under Development ⚠ : Discontinued

Tabelle 12: opsi-local-image

Netboot product	Opsi 4.0.7 / 4.1	Remark
opsi-local-image-prepare	✓	
opsi-local-image-backup	✓	
opsi-local-image-restore	✓	
opsi-local-image-wim-capture	✓	
opsi-local-image-win*	✓	
opsi-local-image-ubuntu	✓	
opsi-local-image-opensuse13-2	⚠	
opsi-local-image-opensusel42-2	🚧	
opsi-vhd-win10-x64	✓	

✓ : Supported ✗ : Unsupported 🚧 : Under Development ⚠ : Discontinued

9.6.2 Vorbedingungen für das Arbeiten mit UEFI / GPT

Dieses Modul ist momentan eine [kofinanzierte opsi Erweiterung](#).

Es sind eine Reihe von Vorbedingungen nötig, um dieses Modul einsetzen zu können. Das bedeutet, das Sie zum Einsatz eine Freischaltdatei benötigen. Diese Freischaltung erhalten Sie wenn Sie die Erweiterung kaufen. Zu Evaluierungszwecken stellen wir Ihnen auch eine zeitlich befristete Freischaltung kostenlos zur Verfügung (→ mail an info@uib.de).

Weitere Details hierzu finden Sie in Abschnitt 9.1.

Technische Voraussetzungen sind opsi 4.0.5 mit den Paketständen:

Tabelle 13: Benötigte Pakete

opsi-Paket	Version
Netbootprodukte	>=4.0.5
opsi Server Pakete	>=4.0.5

9.6.3 Weitere Hinweise für die Verwendung des opsi-Moduls UEFI / GPT

- opsi 4.0.5 unterstützt nur 64-Bit UEFI-Installationen
- Für die Installation über PXE-boot wird ein UEFI-taugliches `winpe_uefi` (analog dem `winpe` für die legacy-Installation) benötigt. Oftmals enthält ein existentes `winpe` bereits UEFI- Unterstützung (Ordner `EFI` und Datei `bootmgr.efi` im `winpe`-Ordner des opsi-Netboot-Produkts.) Ist dies nicht der Fall, sollte ein aktuelles `winpe` mittels `dism` erstellt werden (siehe: "Erstellen eines Winpe" im "Getting-Started - Dokument".) Ein UEFI `winpe` wird im Ordner `winpe_uefi` neben dem `winpe` verzeichnis erwartet. Sofern man ein `winpe` für beide Bootmodi hat, kann man `winpe_uefi` durch einen Soft-Link auf `winpe` ersetzen.
- Einen externen DHCP-Server müssen Sie so konfigurieren, dass er ein PXE-Boot über den opsi-server ermöglicht. Als Bootfile ist einzutragen: `linux/pxelinux.cfg/elilo.efi`
- Im Managementinterface opsi-configed muss für uefi-Clients das Häkchen "Uefi-Boot" gesetzt sein (seit version 4.0.5.8.1) oder aber der Hostparameter `clientconfig.dhcpd.filename=linux/pxelinux.cfg/elilo.efi` Client-spezifisch konfiguriert werden.
- Einstellungen im BIOS:
Da die BIOS-Menüs sehr unterschiedlich sind und unterschiedliche Begrifflichkeiten verwenden, müssen Sie hier überlegen wie das hier gesagte am besten auf Ihr BIOS passt.
 - Secureboot disabled
Dieser Eintrag liegt meistens im Bereich *Boot* oder *Startup* kann aber auch im Bereich *Security* liegen.
 - Schalten Sie das BIOS in den UEFI-Mode. Wenn Sie die Wahl haben zwischen *UEFI only*, *Legacy only*, *Both* dann nehmen Sie *UEFI only*. Wenn der Rechner fest auf *Both* steht ist dies schlecht, kann aber evtl. trotzdem funktionieren. Wenn es *Legacy Support* gibt, so diesen disablen. *CSM Support* in Verbindung mit *UEFI only* kann enabled bleiben., ansonsten eher disablen. *UEFI Network Boot* muss enabled werden. Der entsprechende Eintrag kann auch *Network Stack* heissen und unterhalb von *UEFI* liegen. Wenn dies noch gesondert für *IPv4* und *IPv6* passieren kann ist hier *IPv4* die richtige Wahl.

9.6.4 Einführung

Neue PC's, Tablets und Server enthalten meist ein UEFI BIOS. Häufig kann dieses in einen Legacy Mode umgestellt werden, um das bisherige Verhalten und die Unterstützung eines PXE Boots zu bekommen. Es tauchen aber auch zunehmend Geräte mit UEFI-only BIOS auf (besonders bei den Tablets). Diese lassen sich in einer bisherigen opsi Umgebung nicht per netboot verwalten.

Um auch diese Geräte in opsi integrieren bzw. die Vorteile von UEFI nutzen zu können, entwickelt die uib gmbh die opsi Erweiterung UEFI Support.

9.6.5 Was ist Uefi und was ist hier anders ?

UEFI steht für *Unified Extensible Firmware Interface* und ist der Nachfolger des klassischen PC-BIOS (auch MBR-BIOS genannt).

Für Detaillierte Informationen zum Thema UEFI finden sich unten einige Links.

UEFI ist ungleich mächtiger als das herkömmliche BIOS. Im Prinzip ist UEFI ein eigenes kleines Mini-Betriebssystem. Hier sollen aber nur Punkte erwähnt werden, welche für den Systemverwalter von besonderer Bedeutung sind:

- Die derzeitige (Stand Januar 2014) Umsetzung von UEFI in den BIOSen der Hardware Hersteller ist weit von dem Entfernt was man einen Standard nennen könnte. Vielmehr herrscht ein großes durcheinander wenn von einem anderen Medium geboot werden soll wie der Festplatte. UEFI und klassisches BIOS existieren teilweise Parallel, lassen sich mal einzeln deaktivieren mal nicht. UEFI ist mal mit Compatibility Support Module (CSM) implementiert mal ohne. Netboot geht oder auch nicht. Gerade der letzte Punkt (Netboot) ist natürlich für ein strukturiertes Client Management von großer Bedeutung.
- Im PC-BIOS ist das BIOS und dessen Einstellungen vom OS (in aller Regel) getrennt. D.h. BIOS Einstellungen wie z.B. die Bootreihenfolge sind statisch und können vom Betriebssystem nicht geändert werden. Dies ist bei UEFI anders. Das Betriebssystem kann die Bootreihenfolge ändern (und macht davon in der Regel auch Gebrauch). Auch dies hat wiederum Auswirkungen auf die Anbindung der Rechner an das Clientmanagement durch den Netboot.
- Das UEFI-Bios enthält einen eigenen Bootmanager der nicht nur (siehe oben) in seiner Reihenfolge durch das Betriebssystem veränderbar ist, sondern auch die Starteinträge der Betriebssysteme selbst enthält. Damit wird ein nebeneinander von Unterschiedlichen Betriebssystemen vereinfacht weil die Bootloader unterschiedlicher Betriebssysteme sich nicht mehr überschreiben.
- Das UEFI BIOS ist in 32 oder 64 Bit implementiert. Damit ist dann auch die *Bitigkeit* des Betriebssystems vorgegeben. D.h. kein 32 Bit OS auf einem 64 Bit UEFI System.
- Secureboot (von opsi noch nicht unterstützt)
- Partitionierung mit GPT und zusätzliche Partitionen für die Bootloader:
 - 1. Partition: EFI System Partition (ESP) 100 - 260 MByte ; VFAT
 - 2. Partition: Microsoft Reserved (MSR) 32 - 128 MB; NTFS
 - Ab hier kommen jetzt die eigentlichen OS Partitionen

Links:

[Wikipedia: Unified Extensible Firmware Interface](#)

9.6.6 Was ist anders an GPT

GPT (GUID Partition Table) ersetzen die bisherigen MBR Partitionstabellen. GPT ist Bestandteil der UEFI Spezifikation.

Wesentliche Merkmale für den Sysadmin sind:

- Wegfall der 2 Terabyte Grenze (jetzt 8 Zebibyte)
- *Beliebig* viele primäre Partitionen
- Geänderte Partitionstypen / GUIDs
- Neu: Partitions-GUIDs
- Neu: Partitions-Attribute (Hidden, Read only, ...)
- Andere Werkzeuge zur Bearbeitung: gdisk

Prinzipiell lässt sich GPT auch ohne UEFI verwenden. UEFI funktioniert aber nur mit GPT. Wird UEFI verwendet, so kommen ein bis zwei zusätzliche Partitionen hinzu:

1. Die EFI System Partition (ESP) - hier liegen die Bootloader
2. MS Reserved (MSR)

Links:

- [Wikipedia: GUID Partition Table](#)

9.6.7 UEFI Boot

Im Gegensatz zu herkömmlichen BIOSen kann hier die Bootreihenfolge nicht nur für Geräte, sondern auch für unterschiedliche Bootloader auf der EFI System Partition eingetragen werden. Darüberhinaus ist diese Reihenfolge auch von einem laufenden Betriebssystem manipulierbar. Wenn Sie also den Netboot als oberste Priorität festlegen, so wird dies die erste OS-Installation nicht überleben.

9.6.8 UEFI Netboot

Leider unterstützen viele frühe UEFI BIOSe noch kein Netboot, aber die Unterstützung wird zunehmend besser.

Es gibt inzwischen schon eine ganze Reihe von UEFI Bootloadern (wie z.B. grub2, elilo), leider unterstützen die wenigsten davon einen Netboot.

Die uib gmbh stellt mit der opsi-Erweiterung UEFI Support eine funktionierende Unterstützung für die Anbindung eines Clients über einen UEFI-Netboot an opsi vor. Gleichzeitig erwarten wir, dass im Rahmen der technischen Weiterentwicklung diese opsi-Erweiterung in den nächsten Jahren noch deutlich umgebaut werden wird.

9.6.9 Opsi Unterstützung für UEFI Netboot

Die opsi Unterstützung für UEFI ist im Rahmen einer Reihe von Teilkomponenten umgesetzt:

- Anpassung des netbootfähigen UEFI Bootloaders ELILO an die opsi / Client-Management Bedürfnisse.
- Neuer opsipxeconfd, welcher neben Konfigurationsdateien für das bisherige PXE nun auch Konfigurationsdateien für den opsi-ELILO bereitstellt.
- Bereitstellung neuer (64Bit) opsi-linux-bootimages mit den Werkzeugen für UEFI- und GPT-Management
- Umbau aller Netbootprodukte zur Betriebssysteminstallation (Windows/Linux) auf die zusätzliche Unterstützung von UEFI/GPT. Hiervon sind natürlich jene Betriebssysteme ausgenommen welche von sich aus keine UEFI Unterstützung haben.
- Speicherung der Einstellung ob ein Client als UEFI Client behandelt werden soll auf dem opsi-server (clientconfig.dhcpd.filename=linux/pxelinux.cfg/elilo.efi)

- Unterstützung einer Software gesteuerten Umschaltung auf UEFI-Netboot.
Soweit das BIOS dies ermöglicht (es also einen aktivierbaren Netbooteintrag im Bios gibt) wird das Label des UEFI-Netboot Eintrages des Bios auf dem opsi-server gespeichert (clientconfig.uefinetbootlabel). Dies ermöglicht es opsi-produkten gezielt für den nächsten boot das Bios auf Netboot umzustellen. Diese Technik ist in verschiedenen opsi-Produkten implementiert. Ein wichtiges Beispiel ist das Produkt **opsi-uefi-netboot**:
Dieses Produkt versucht das Bios auf netboot umzustellen und dann einen reboot auszulösen. Wird kein gespeichertes `uefinetbootlabel` gefunden oder handelt es sich nicht um einen UEFI Rechner wird nur ein reboot ausgelöst. Diese Produkt funktioniert sowohl unter Windows wie auch unter Linux.

9.6.10 Installation

Alle notwendigen Pakete sind ab der opsi Version 4.0.5 automatisch installiert.

9.6.11 Konfiguration des DHCP Servers

Damit Uefi-Clients auch per PXE-Boot betankt werden können, benötigen sie einen anderen Bootfile Eintrag, als den herkömmlichen für Bios-Clients. Als Bootfile ist einzutragen: *linux/pxelinux.cfg/elilo-x86.efi*. Da in der Praxis vermutlich beide Varianten in einer opsi-Umgebung betrieben werden sollen, muss der DHCP-Server je nach Clienttyp das entsprechende Bootfile auf dem opsi-Server zuweisen. Hierzu folgen Beispielkonfigurationen für DHCP-Server auf Basis von Linux- oder Windows-Systemen.

Beispiel aus einer Konfiguration eines Linux isc-dhcp-server

Auf einem Linux-Server kann in die `dhcpd.conf` folgende Weiche konfiguriert werden:

```
filename "linux/pxelinux.0";

# Das ist die UEFI Detection:
if substring (option vendor-class-identifier , 19,1 ) = "0" {
    log (info , "pxe client");
    filename "linux/pxelinux.0";
}
else if substring (option vendor-class-identifier , 19,1 ) = "6" {
    log (info , "efi32 client");
    filename "linux/pxelinux.cfg/elilo-x86.efi ";
}
else if substring (option vendor-class-identifier , 19,1 ) = "7" {
    log (info , "efi64 client");
    filename "linux/pxelinux.cfg/elilo.efi ";
}
else {
    log (info , concat ( "Unhandled vendor class Arch: ", substring (option
    vendor-class-identifier , 19,1 )));
}
}
```

Siehe auch: [Fedora: PXE Boot Configuration](#)

Beispiel für die Konfiguration auf einem Windows DHCP-Server 2012 R2

- Als Standard wird in dieser Variante der PXE-Bootfile für Uefi 64Bit Installationen eingetragen. Dafür werden die DHCP-Optionen 66 und 67 wie folgt angepasst:
066 Hostname des Startservers: <IP des Opsiservers>
067 Name der Startdatei: linux/pxelinux.cfg/elilo.efi

- Zur Unterscheidung der Bios-Clients muss auf dem DHCP-Server ein Herstellerklasse-Identifizier (PXEClient:Arch:00000:UNDI:002001) definiert werden:

```

Herstellerklasse definieren
Neue Herstellerklasse hinzufügen
Klasse bearbeiten
    Anzeigename: Legacy BIOS
    Ascii: PXEClient:Arch:00000:UNDI:002001

```

- Die vordefinierten Optionen müssen dem Herstellerklasse-Identifizier zugeordnet werden:

```

Vordefinierte Optionen einstellen
Optionen hinzufügen
    Optionsklasse: Legacy BIOS
    Hinzufügen
Optionstyp anpassen
    Name: Legacy BIOS
    Datentyp: Zeichenfolge
    Code: 60
    Beschreibung: PXEClient Class Legacy BIOS
Vordefinierte Optionen und Werte
    Zeichenfolge: PXEClient

```

- DHCP-Richtlinie definieren, die den Bootfile für PXE-Boot(BIOS) der Herstellerklasse zuordnet:

```

Neue Richtlinie
    Richtlinienname: PXE BootFile Legacy BIOS
    weiter
Bedingungen hinzufügen
    Kriterien: Herstellerklasse
    Operator: ist gleich
    Wert: Legacy BIOS
    hinzufügen
Möchten Sie einen IP-Adressbereich für die folgende Richtlinie
konfigurieren: Nein
Herstellerklasse: DHCP Standard Options
    067 Name der Startdatei
    Dateieingabe
    Zeichenfolgenwert: linux/pxelinux.0

```

- In den Bereichsoptionen gibt es dann zwei Einträge für die Startdatei, die in einem Fall zur Erkennung eines Bios Clients an eine Richtlinie gekoppelt ist:

067 Name der Startdatei: linux/pxelinux.cfg/elilo.efi	Richtlinie: Keine
067 Name der Startdatei: linux/pxelinux.0	Richtlinie: PXE BootFile Legacy BIOS

Referenz: [Preboot Execution Environment \(PXE\) Specification](#)

9.6.12 Konfiguration des opsipxeconfd

Seit opsipxeconfd 4.0.7.7 ist es möglich den Pfad der als Vorlage verwendeten Dateien über die Konfigurationsdatei `opsipxeconfd.conf` anzupassen.

Dazu können in dieser Datei über die Optionen `uefi netboot config template x86` bzw. `uefi netboot config template x64` die Pfade zu den entsprechenden Dateien angegeben werden.

9.6.13 Alternative elilo.uefi

Die standardmäßig verteilte `elilo.efi` ist so konfiguriert, dass unendlich lange auf Informationen für einen Netboot gewartet wird. In den meisten Szenarien ist das kein Problem, da nach einer erfolgreichen Installation die Bootreihenfolge auf einen Boot von Festplatte angepasst wird. Es gibt allerdings einige Geräte, bei denen diese Anpassung nicht automatisiert möglich ist, obwohl dies in der UEFI-Spezifikation vorgeschrieben ist.

Es gibt eine alternative `elilo.efi`, welche nur eine kurze Zeit wartet, bevor ein Boot von Festplatte gemacht wird. Leider funktioniert dies nicht mit allen Geräten, insbesondere bei der Verwendung von NVME-Festplatten, und ist daher nicht der Default.

Falls Sie diese Alternative `elilo.efi` verwenden wollen, so ist sie unter https://download.uib.de/opsi4.1/misc/uefi/elilo_with_timeout/ zu finden und kann dann im tftpboot-Verzeichnis plaziert werden (in der Regel `/tftpboot/linux/pxelinux.cfg/`), um das Original zu ersetzen.

9.6.14 Kriterien für ein gutes BIOS

Ob ein UEFI BIOS für die Anforderungen eines Client-Management-Systems wie opsi geeignet ist, hängt von verschiedenen Kriterien ab. Diese Kriterien sagen nichts über die Qualität des Gerätes als solches aus, sondern nur über seine Wartbarkeit per Netboot-Anbindung. Es geht hier also um die BIOS-Funktionen zum UEFI Netboot. Hier ein beispielhafter Vergleich :

Tabelle 14: Beispiele für UEFI BIOS Unterschiede

	Lenovo Twist	MS-Surface	Dell Venue 11
UEFI pur	✓	✓	✓
UEFI + CSM	✓	x	✓
Legacy	✓	x	✓
Both	✓	x	x
UEFI Netboot	✓	✓	✓
Aktivierbarer Eintrag	✓	x	✓
Netboot ohne Interaktion	✓	x	✓

Unter *Aktivierbarer Eintrag* verstehen wir, das sich per Standard-Software ein Netboot für den nächsten Reboot aktivieren läßt. *Netboot ohne Interaktion* bedeutet, dass ein aktivierter Netboot-Eintrag beim Reboot ausgeführt wird und dazu keine Interaktion (drücken von Tastenkombinationen, F12-Taste, ...) nötig sind. Sind diese Voraussetzungen gegeben, können bestimmte opsi Produkte einen Netboot auslösen. Dies ist für die Automatisierung von Abläufen von großer Bedeutung. Ein Produkt in dem dies implementiert ist ist das Localbootprodukt für Windows und Linux `opsi-uefi-netboot`.

9.6.15 Technisches

Die folgenden Unterkapitel dienen als Wissensbasis zu dem händigen oder gescrripteten Umgang mit UEFI / GPT. Zum Verständnis wie opsi mit UEFI/GPT arbeitet sind sie nicht erforderlich.

Technisches zu UEFI

Manipulation der UEFI Bootloader Einträge passiert unter Linux mit dem Programm `efibootmgr`.

Liste der Booteinträge:

```
efibootmgr -v
BootCurrent: 000D
Timeout: 0 seconds
BootOrder: 0012,0011,000D,0010,000B,0009,0007,0008,000A,000C
Boot0000 Setup
Boot0001 Boot Menu
(..)
Boot0007* USB CD          030a2400d23878bc820f604d8316c068ee79d25b86701296aa5a7848b66cd49dd3ba6a55
Boot0008* USB FDD        030a2400d23878bc820f604d8316c068ee79d25b6ff015a28830b543a8b8641009461e49
Boot0009* ATA HDDO       030a2500d23878bc820f604d8316c068ee79d25b91af625956449f41a7b91f4f892ab0f600
Boot000D* PCI LAN        030a2400d23878bc820f604d8316c068ee79d25b78a84aaf2b2afc4ea79cf5cc8f3d3803
Boot0010* ubuntu         HD(1,800,31801,faffb7b9-bdf9-4767-b475-0b8aee68d3ac)File(\EFI\ubuntu\grubx64.efi)
Boot0011* opsitempwinpe  HD(4,3c72800,7cf801,dc1cea68-a296-4fb8-a97a-263227ed86f4)File(\EFI\boot\bootx64.efi)
Boot0012* Windows Boot Manager HD(1,800,31801,5e4ffde2-3e25-42dd-b0f7-fcb7ee5d2b20)File(\EFI\Microsoft\Boot\bootmgfw.\efi)WINDOWS.....x...B.C.D.O.B.J.E.C.T.=.{9.d.e.a.8.6.2.c.-.5.c.d.d.-.4.e.7.0.-.a.c.c.1.-.f.3.2.b.3.4.4.d\
.4.7.9.5.}...a.....-...
```

Manipulation der UEFI Bootloader Einträge passiert unter Windows mit dem Programm `bcdedit`.

Liste der Booteinträge:

```
bcdedit /enum firmware

Start-Manager für Firmware
-----
Bezeichner          {fwbootmgr}
displayorder        {bootmgr}
                    {99a9f9be-9a98-11e3-b22f-806e6f6e6963}
                    {11a8b97e-99df-11e3-ae5c-b888e3e3cbb4}
                    {11a8b986-99df-11e3-ae5c-b888e3e3cbb4}

Windows-Start-Manager
-----
Bezeichner          {bootmgr}
device              partition=\Device\HarddiskVolume1
path                \EFI\Microsoft\Boot\bootmgfw.efi
Firmwareanwendung (101fffff)
-----
Bezeichner          {11a8b971-99df-11e3-ae5c-b888e3e3cbb4}
description         Setup
Firmwareanwendung (101fffff)
-----
Bezeichner          {11a8b972-99df-11e3-ae5c-b888e3e3cbb4}
description         Boot Menu
(...)
Firmwareanwendung (101fffff)
-----
Bezeichner          {11a8b978-99df-11e3-ae5c-b888e3e3cbb4}
description         USB CD
Firmwareanwendung (101fffff)
-----
Bezeichner          {11a8b979-99df-11e3-ae5c-b888e3e3cbb4}
description         USB FDD
Firmwareanwendung (101fffff)
-----
Bezeichner          {11a8b97a-99df-11e3-ae5c-b888e3e3cbb4}
description         ATA HDDO
Firmwareanwendung (101fffff)
-----
Bezeichner          {11a8b97e-99df-11e3-ae5c-b888e3e3cbb4}
description         PCI LAN
Firmwareanwendung (101fffff)
-----
Bezeichner          {99a9f9be-9a98-11e3-b22f-806e6f6e6963}
```

```
device      partition=X:
path        \EFI\boot\bootx64.efi
description opsitepwinpe
```

Mit beiden Programmen lassen sich Einträge erstellen, löschen, der *nextboot* setzen und die Reihenfolge manipulieren. Beispiel: Eintrag für den nächsten Boot setzen:

- Linux:

```
efibootmgr /bootnext <hexId>
```

- Windows:

```
bcdedit /set {fwbootmgr} bootsequence <GUID>
```

Technisches GPT

GPT Partitionen kennen *neue* Partitionstypen. Diese sind an die bisher bekannten angelehnt. So wird aus dem Partitionstyp für NTFS 07 unter GPT 0700. Analog wird aus den Linux Partitionstypen 82 und 83 entsprechend 8200 und 8300.

Die Liste der bekannten Partitionstypen kann man sich anzeigen lassen:

```
# sgdisk -L
0700 Microsoft basic data 0c01 Microsoft reserved 2700 Windows RE
4100 PowerPC PReP boot 4200 Windows LDM data 4201 Windows LDM metadata
7501 IBM GPFS 7f00 ChromeOS kernel 7f01 ChromeOS root
7f02 ChromeOS reserved 8200 Linux swap 8300 Linux filesystem
8301 Linux reserved 8302 Linux /home 8400 Intel Rapid Start
8e00 Linux LVM a500 FreeBSD disklabel a501 FreeBSD boot
a502 FreeBSD swap a503 FreeBSD UFS a504 FreeBSD ZFS
a505 FreeBSD Vinum/RAID a580 Midnight BSD data a581 Midnight BSD boot
a582 Midnight BSD swap a583 Midnight BSD UFS a584 Midnight BSD ZFS
a585 Midnight BSD Vinum a800 Apple UFS a901 NetBSD swap
a902 NetBSD FFS a903 NetBSD LFS a904 NetBSD concatenated
a905 NetBSD encrypted a906 NetBSD RAID ab00 Apple boot
af00 Apple HFS/HFS+ af01 Apple RAID af02 Apple RAID offline
af03 Apple label af04 AppleTV recovery af05 Apple Core Storage
be00 Solaris boot bf00 Solaris root bf01 Solaris /usr & Mac Z
bf02 Solaris swap bf03 Solaris backup bf04 Solaris /var
bf05 Solaris /home bf06 Solaris alternate se bf07 Solaris Reserved 1
bf08 Solaris Reserved 2 bf09 Solaris Reserved 3 bf0a Solaris Reserved 4
bf0b Solaris Reserved 5 c001 HP-UX data c002 HP-UX service
ea00 Freedesktop $BOOT eb00 Haiku BFS ed00 Sony system partitio
ef00 EFI System ef01 MBR partition scheme ef02 BIOS boot partition
fb00 VMWare VMFS fb01 VMWare reserved fc00 VMWare kcore crash p
fd00 Linux RAID
```

Tatsächlich sind diese hier gezeigten Partitionstypen nur *Abkürzungen* für die eigentlich verwendeten GUID's welche dem Partitionierungsschema den Namen gegeben haben.

Also: 0700 steht für Microsoft basic data und für die GUID EBD0A0A2-B9E5-4433-87C0-68B6B72699C7

Eine Liste der GUID's findet sich z.B. bei Wikipedia:

https://de.wikipedia.org/wiki/GUID_Partition_Table#Partitionstyp-GUIDs

https://en.wikipedia.org/wiki/GUID_Partition_Table#Partition_type_GUIDs

Weiterhin kennt das Werkzeug gdisk (und sgdisk, ...) eine interne Ersetzungstabelle für unbekannte Partitionstypen. So gibt es für den *alten* Partionstyp für vfat32 0b keine Entsprechung als 0b00. Übergibt man aber trotzdem 0b00 als Typ an sgdisk so wird ohne jede Meldung hieraus 0700. Dies passiert wohl nach der Überlegung: vfat32 - das wird schon eine Microsoft Daten Partition sein...

GPT Partitionen kennen Attribute.

Liste der z.Zt. bekannten Attribute

Wert	Beschreibung	Attributwert (sgdisk --info / diskpart gpt attribute)
nix	nix	0000000000000000
0	Systempartition (system partition)	0000000000000001
1	Verstecke die Partition vor EFI (hide from EFI)	0000000000000002
2	Legacy Bootflag (legacy BIOS bootable)	0000000000000004
60	Nur lesen (read-only)	1000000000000000
62	Versteckt (hidden)	4000000000000000
63	Nicht Einhängen (do not automount)	8000000000000000

Die attribute werden unter Linux bei sgdisk über die Option `-A`, `--attributes` unter Verwendung der Kurzform und unter Windows über diskpart mit dem Befehl `gpt attributes` unter Verwendung der Langform gesetzt.

Beispiele:

```
select disk 0
select partition 1
gpt attributes=0x0000000000000000
```

```
sgdisk -t 1:0700 --attributes 1:clear:63 --attributes 1:set:62 -p /dev/sda
```

Anzeigen der Partitionstabelle mit `-p` , `--print`:

```
sgdisk -p /dev/sda
```

Anzeigen der Detailinfos zu einer Partition (1) mit `--info=`:

```
sgdisk --info=1 /dev/sda
```

opsi UEFI/GPT Roadmap

- UEFI 32 Bit Unterstützung
- Andere netbootfähige UEFI Bootloader (grub2)
- Secureboot

9.7 opsi local image

9.7.1 Vorbedingungen für die opsi Erweiterung opsi local image

Dieses Modul ist momentan eine [kofinanzierte opsi Erweiterung](#).

Es sind eine Reihe von Vorbedingungen nötig, um dieses Modul einsetzen zu können. Das bedeutet, dass Sie zum Einsatz eine Freischaltdatei benötigen. Diese Freischaltung erhalten Sie, wenn Sie die Erweiterung kaufen. Zu Evaluierungszwecken stellen wir Ihnen auch eine zeitlich befristete Freischaltung kostenlos zur Verfügung (→ mail

an info@uib.de).

Weitere Details hierzu finden Sie in Abschnitt 9.1.

Mit der Berechtigung opsi-local-image zu verwenden erwerben Sie gleichzeitig auch das Recht die Erweiterung opsi-vhd-reset (siehe Abschnitt 9.8) zu verwenden.

Technische Voraussetzungen sind opsi 4.0.3 mit den Paketständen:

Tabelle 15: Benötigte Pakete

opsi-Paket	Version
opsi-linux-bootimage	>= 20130207-1



Achtung

Für das Produkt opsi-local-image-capture muß der share opsi_depot_rw für *pcpatch* beschreibbar sein. Prüfen Sie Ihre Samba Konfiguration.

9.7.2 Einführung

Opsi bietet eine gute Basis, um automatisiert Windows Rechner zu installieren und zu pflegen - auch und gerade, wenn es sich um heterogene Hardware handelt. Die von opsi unterstützte Technik der Paket basierten Installation ist aber nicht schnell genug, um in Schulungsräumen Rechner innerhalb von kurzer Zeit wie z.B. in einer Pause zwischen zwei Kursen wieder in einen definierten Zustand zu versetzen. Daher wird hier ein Konzept vorgestellt, bei dem die paketbasiert durchgeführte Installation lokal auf einer zweiten Partition als Imagekopie gesichert wird und von dort eine schnelle Wiederherstellung möglich ist.

1. Initiale Installation mit abschließender lokaler Image-Sicherung
2. Schnelle Wiederherstellung auf Basis unterschiedlicher Techniken
3. Systempflege mit abschließender lokaler Image-Sicherung
4. Integration von Capturing von vorhandenen Installationen zu WIM
5. Integration von Linuxclients in das Backup/Restore Verfahren.

9.7.3 Konzept

Die Anforderungen edukativer Computernetze unterscheiden sich teilweise von denen anderer Netzwerke. Eine wesentliche Anforderung, die im Folgenden diskutiert wird, ist die schnelle Wiederherstellung von Rechnern in einen definierten Zustand, den Sie im Rahmen einer temporären Nutzung verloren haben. Konkret geht es um die Bereitstellung von Rechnern in schulischen Computerräumen, wobei die Problemlage auch auf kommerzielle Computerräume oder universitäre Computerpools zutrifft.

Die Wiederherstellung soll innerhalb von kurzer Zeit (ca. 15 Minuten) erfolgen und sollte soweit möglich einen Rechner nicht nur *zurücksetzen* sondern auf eine andere Basisinstallation (z.B. Win XP / Win 7 / Linux) umschalten können. Weiterhin muss es möglich sein, dass eine kontinuierliche Pflege der Systeme mit Sicherheitsupdates gewährleistet ist.

Die Üblichen Techiken zur Installation von PC's haben verschiedene spezifische Vor- und Nachteile:

Tabelle 16: Vor- und Nachteile von Unattend und Image Lösungen

Feature	Unattend	Image
Geschwindigkeit	(-) langsam	(+) schnell
Empfindlichkeit gegen heterogene Hardware	(+) gering	(-) hoch
Netzwerkbelastung	(-) hoch	(-) hoch

Das Konzept von opsi-local-image versucht die Vorteile beider klassischen Konzepte miteinander zu kombinieren:

Tabelle 17: opsi-local-image

Feature	Unattend
Geschwindigkeit	(+) schnell
Empfindlichkeit gegen heterogene Hardware	(+) gering
Netzwerkbelastung	(+) gering

Das Konzept kann in den folgenden Stichpunkten zusammengefasst werden:

- Initiale Windowsinstallation per PXE-Boot paketbasiert mit individueller Treiberintegration unter Verwendung des opsi-Linux-Bootimage
- Sicherung dieser initialen Installation in einem Sicherungsbild auf einer 2. Partition auf der lokalen Festplatte des Rechners unter Verwendung des opsi-Linux-Bootimage
- Schnelle Wiederherstellung der Produktiv-Installation aus dem lokalen Image unter Verwendung des opsi-Linux-Bootimage
- Pflege der lokalen Installation (Sicherheitsupdates) über die opsi Softwareverteilung und Sicherung des aktualisierten Systems auf das lokale Backup-Image unter Verwendung des opsi-Linux-Bootimage

9.7.4 Technisches Konzept

Der Schulungsrechner wird mit einer statischen Partitionstabelle verwendet. Dabei kann entweder mit 3 oder 4 Partitionen gearbeitet werden:

- Partition 1 (System)
Hier findet sich das aktuell verwendete Betriebssystem (Windows / Linux)
Die Größe dieser Partition wird bei der Partitionierung über das Produkt `opsi-local-image-prepare` über ein Property gesteuert.
- Optional: Partition 2 (sysdata)
Hier können z.B. Anwenderdaten gespeichert werden, welche bei einem Restore nicht überschrieben werden. Die Formatierung ist NTFS.
Die Größe dieser Partition wird bei der Partitionierung über das Produkt `opsi-local-image-prepare` über ein Property gesteuert.
- Partition 3 (winpe / swap)
Die Größe dieser Partition ist statisch auf 4GB festgelegt.
Unter Windows XP wird diese Partition nicht verwendet.
Unter NT6 (Windows 7) wird diese Partition für das bei der Installation notwendige winpe verwendet und ist im Betrieb nicht sichtbar.
Unter Linux wird diese Partition als Swap verwendet.

- Partition 4 (backup)
Diese Partition dient zur Speicherung der gesicherten Images und ihrer Metadaten.
Die Größe dieser Partition ergibt sich aus dem nach Erstellung der anderen Partitionen noch vorhandenen freien Platz.

Die Netbootprodukte zur Betriebssystem Installation verwenden nur die ersten zwei oder drei Partitionen (XP nur die erste) und lassen die letzte Backup-Partition unberührt. Somit bleiben die auf der Partition 4 (backup) liegenden Images auch bei der Installation eines neuen Betriebssystems erhalten.

9.7.5 Abläufe

Initiale Installation

Über das Produkt `opsi-local-image-prepare` wird zunächst die notwendige statische Partitionierung erzeugt.

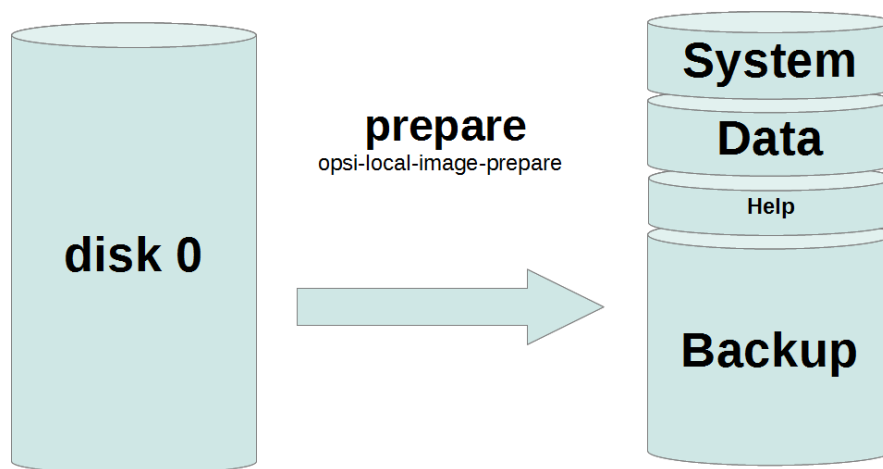


Abbildung 90: Schema: Statische Partitionierung mit opsi-local-image-prepare

Anschließend können über die Produkte `opsi-local-image-win*` und andere verschiedene Betriebssysteme mit unterschiedlicher Ausstattung an Anwendungssoftware installiert werden.

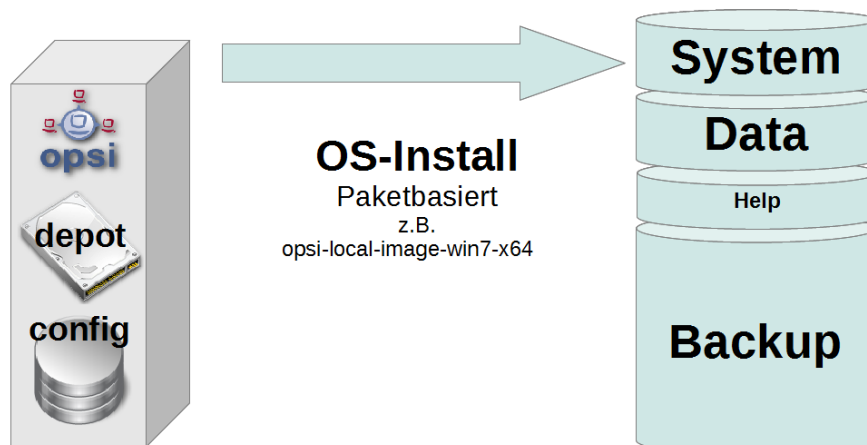


Abbildung 91: Schema: OS Installation mit opsi-local-image-win*

Diese werden per default nach Ihrer Installation automatisch als Image gesichert.

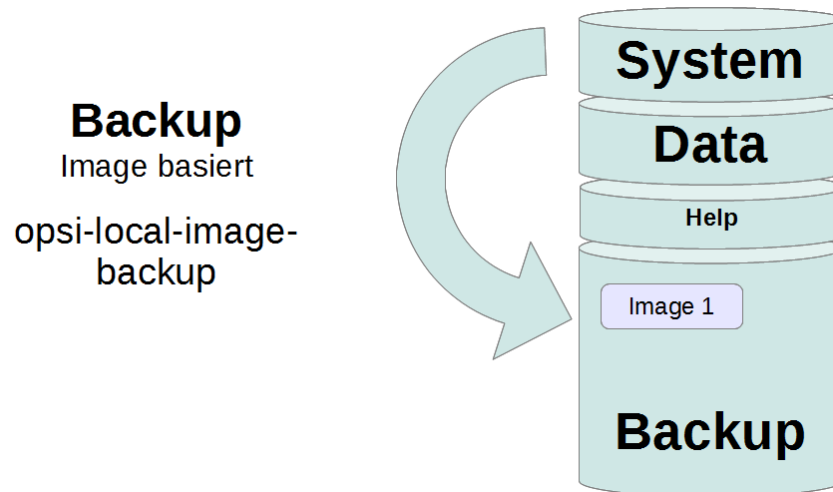


Abbildung 92: Schema: Image Backup mit opsi-local-image-backup

Restore eines Images

Der einfache Aufruf des Produktes `opsi-local-image-restore` stellt automatisch das letzte erstellte Image wieder her. Soll ein anderes Image wieder hergestellt werden, so muß dies im Property `imagefile` angegeben werden.

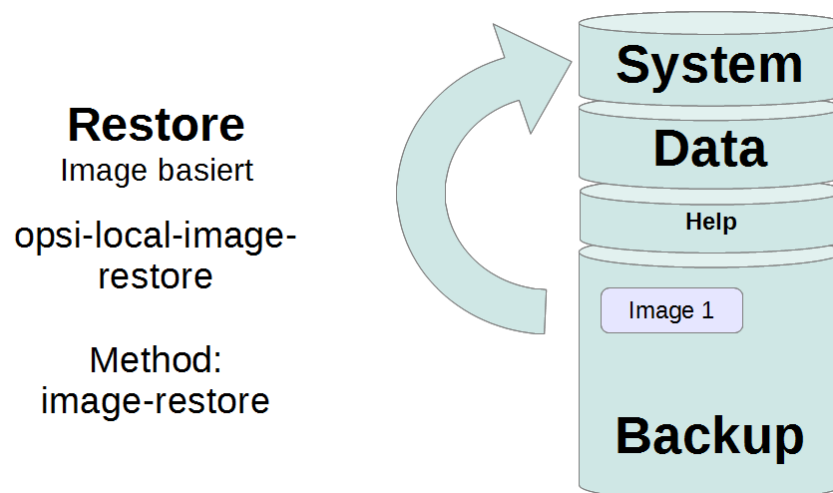


Abbildung 93: Schema: Image Restore mit opsi-local-image-restore

Löschen eines Images

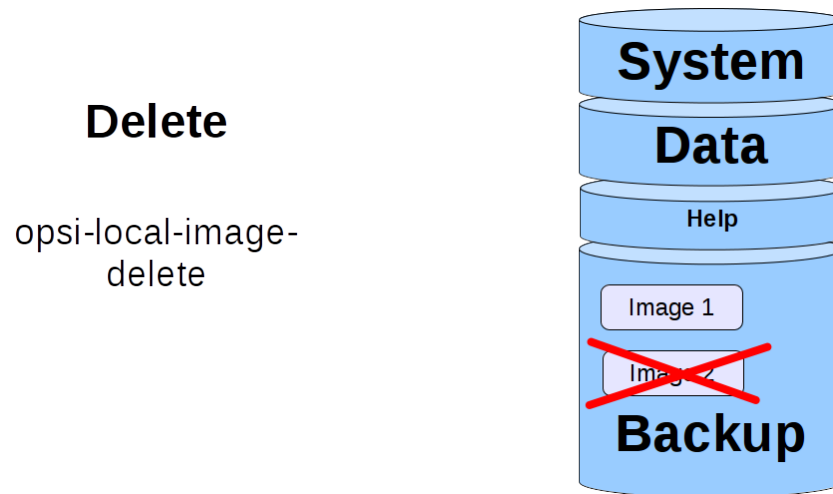


Abbildung 94: Schema: Löschen eines gespeicherten Images

Durch den Aufruf des Produktes `opsi-local-image-delete` wird das im Property `imagefile` angegebene Image gelöscht.

9.7.6 Update eines Images: Automatisierter Workflow

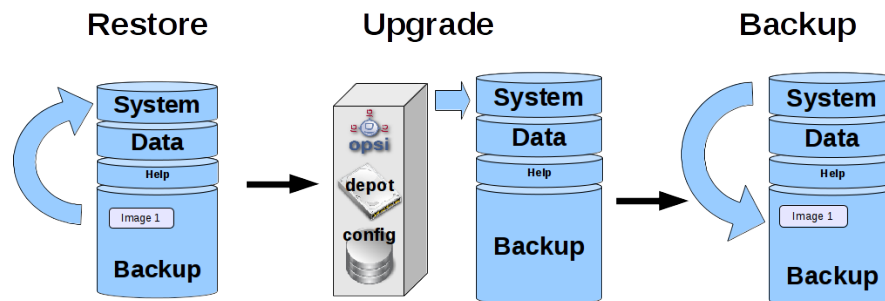


Abbildung 95: Schema: Ablauf des automatischen Upgrades eines gespeicherten Images

Durch den Aufruf des Produktes `opsi-local-image-restore` mit der Propertyeinstellung `update_and_backup = true` wird:

1. das angegebene Image (inklusive seiner opsi Meta-Daten) restored,
2. anhand der restorten Meta-Daten festgestellt welche Produkt per opsi Upgedatet werden können
3. die Updates werden durchgeführt
4. Von der aktualisierten Systempartition wird wieder durch `backup` ein Image erzeugt.

9.7.7 Die opsi-local-image Produkte

Die opsi-local-image Produkte ab Version 4.1 unterstützen auch Mehrplattensysteme. Bitte beachte Sie dazu auch den Abschnitt [Abschnitt 8.2.3](#)

Das Paket *opsi-local-image* besteht aus folgenden Produkten

Das Netbootprodukt zur Partitionierung

- `opsi-local-image-prepare`

Die Netbootprodukte zur Betriebssystem Installation:

- `opsi-local-image-winxp`
- `opsi-local-image-win7`
- `opsi-local-image-win7-x64`
- `opsi-local-image-win81`
- `opsi-local-image-win81-x64`
- `opsi-local-image-win10`
- `opsi-local-image-win10-x64`
- `opsi-local-image-ubuntu`

Den Netbootprodukten zum Handling der lokalen Images

- `opsi-local-image-backup`
- `opsi-local-image-restore`
- `opsi-local-image-delete`

Dem Lokalbootprodukt zur Ablaufsteuerung:

- `opsi-local-image-backup-starter`

Zur Installation der Produkte stellen Sie bitte in der Datei `/etc/opsi/opsi-product-updater.conf` das Attribut **active** des Repositories **uib_local_image** auf *True*. Anschließend können Sie über einen Aufruf von `opsi-product-updater -i -vv` die neuen Produkte installieren lassen.

UEFI Kompatibilität

Die `opsi-local-image` Produkte sind UEFI kompatibel mit wenigen Ausnahmen.

Nicht UEFI kompatibel sind die Produkte:

Netboot Produkt zur Partitionierung

- `opsi-local-image-prepare`
Erstellung der statischen Partitionierung der Festplatte für alle anderen Produkte.
Properties:

ask_before_inst

Soll am Client vor dem Start bestätigt werden. (Default=*true*)

system_partition_size

Gibt die Größe der Partition 1 (system) an (Default = 30G)

data_partition_size

Gibt die Größe der Partition 2 (data) an. Bei eine Größe von *0G* wird keine Partition für Daten angelegt. (Default = 0G)

start_os_installation

Hier kann das Produkt zur Installation eines Betriebssystems ausgewählt werden, welches im Anschluß an die Partitionierung automatisch gestartet wird. Bei der Installation dieses Produkts werden beim Produkt `opsi-local-image-restore` die Produktproperties `imagefile` und `imagefiles_list` gelöscht, da durch die Neupartitionierung diese Daten ungültig geworden sind.

delay_for_reboot

Sekunden zwischen Beendigung des Scriptes und dem Reboot um dem Server Zeit zugeben die Netboot-Pipe zu erstellen.

minimal_backup_partition_size

Diese Property dient zu einer Plausibilitätsüberprüfung der gemachten Angaben.

Die Größe der Backuppartition ergibt sich aus:

Größe der Festplatte - (`system_partition_size` + `data_partition_size` + `winpe_partition_size`).

Üblicherweise verwendet man `opsi-local-image`, weil ein lokales Backup der Systempartition gemacht werden soll. Dazu ist Voraussetzung, das für die Backuppartition genügend Platz ist. Wird bei der Berechnung der Partitionierung festgestellt das der verbleibende Platz für die Backuppartition kleiner ist als der Wert von `minimal_backup_partition_size` wird mit einem Fehler abgebrochen.

(Default=55%)

winpe_partition_size

Größe der winpe Partition (Default=4G)

multi_disk_mode

Diese Property dient zur Wahl der Festplatte auf die installiert werden soll.

Mögliche Werte sind: "0", "1", "2", "3", "prefer_ssd", "prefer_rotational"

Die Werte "0", "1", "2", "3" geben den Index der Festplatte direkt an ("0" = 1. Festplatte)

Der Wert "prefer_ssd" wählt die erste SSD Platte aus.

Der Wert "prefer_rotational" wählt die erste klassische Platte (mit rotierenden Scheiben) aus.

Das Property wird auf Systemen mit nur einer Platte ignoriert.

Default = "0"

backup_partition_on_same_disk

true : Backuppartition auf der Systemplatte anlegen. false : Backuppartition auf der ersten freien Platte anlegen, welche nicht die Systemplatte ist.

Das Property wird auf Systemen mit nur einer Platte ignoriert.

Default = "true"

**Wichtig**

Verwenden Sie dieses Produkt nur zur initialen Vorbereitung der Platte. Es löscht alle gespeicherten Images.

Netboot Produkte zur Installation von Windows

Die Netbootprodukte zur Installation von Windows sind Abkömmlinge der opsi Standardprodukte zur Windowsinstallation. Das bedeutet konkret, dass sie bezüglich des Aufbaus und der Treiberintegration identisch sind. Entsprechende Anleitungen finden sich im *opsi-getting-started* Handbuch.

Die Properties der Windows NT6 Produkte ab Version 4.1 sind eine Teilmenge der Properties der NT6 Standardprodukte wie sie im Abschnitt Abschnitt 8.2.3 beschrieben sind. Die fehlenden Properties zu Platten und Partitionen werden dem produkt `opsi-local-image-prepare` entnommen.

**Wichtig**

Ändern Sie die Propertyeinstellungen von `opsi-local-image-prepare` nicht mehr nach dem sie die Maschine damit präpariert haben, da die nachfolgenden Produkte auf diese Werte zugreifen.

- **opsi-local-image-winxp**
Installation von Windows XP. Verwendet nur die erste Partition. Administrator Passwort ist leer.
- **opsi-local-image-win7**
Installation von Windows 7 32 Bit.
- **opsi-local-image-win7-x64**
Installation von Windows 7 64 Bit.
- **opsi-local-image-win81**
Installation von Windows 8.1 32 Bit.
- **opsi-local-image-win81-x64**
Installation von Windows 8.1 64 Bit.
- **opsi-local-image-win10**
Installation von Windows 10 32 Bit.
- **opsi-local-image-win10-x64**
Installation von Windows 10 64 Bit.

Diese Produkte haben folgende *opsi-local-image* spezifische Properties:

- *backup_after_install* mit dem default Wert *true*. Dies bedeutet in diesem Fall, dass nach der OS Installation zunächst die Anwendungssoftware installiert wird und abschließend eine Imagesicherung der Installation durchgeführt wird. Weiterhin wird der Wert vom *imageFile* des Produktes *opsi-local-image-restore* gelöscht. Dies führt dazu, dass das erstellte Backup den Namen des laufenden Netbootproduktes (z.B. *opsi-local-image-win7*) bekommt.
- *setup_after_install*
Hier können ein oder mehrere Produkte angegeben werden, welche nach Abschluß der Betriebssysteminstallation auf *setup* gestellt werden. Dabei werden auch die Abhängigkeiten diese Produkte mit aufgelöst.

Netboot Produkte zur Installation von Linux

- **opsi-local-image-ubuntu**
Installation von Ubuntu Linux 12.04/14.04 32Bit/64Bit.
Das installierte System kennt 2 user: *root* und *user*. Für *root* wird das im Produktproperty *root_password* festgelegte Passwort gesetzt (Default: *linux123*). Für *user* wird das im Produktproperty *user_password* festgelegte Passwort gesetzt (Default: *linux123*). Details der Installation werden über Produktproperties gesteuert. Die wesentlichen Produktproperties sind dabei:
 - **askbeforeinst:**
Soll das Starten der Installation am Client bestätigt werden müssen? (Default=*true*)
 - **architecture:**
Architektur Auswahl, beeinflusst die Auswahl des bootimages und die Installationsarchitektur. (Default=*64bit*)
 - **additional_packages:**
Welche zusätzlichen Pakete sollen installiert werden? Angabe der Pakete Leerzeichen separiert. (Default = *"*)
 - **language:**
Welche Sprache / locale soll installiert werden. (Default=*de*)
 - **console_keymap**
steuert in welches Tastaturlayout installiert wird. (Default=*de-latin1-nodeadkeys*)
 - **timezone:**
Welche Zeitzone soll verwendet werden?. (Default=*Europe/Berlin*)
 - **online_repository**
gibt an aus welchem online Repository die Pakete installiert werden sollen. Der Default ist *http://de.archive.ubuntu.com/ubuntu*

- **proxy:**
Proxystring (wenn benötigt) in der Form: `http://<ip>:<port>`. (Default = "")
- **backup_after_install**
(*true/false*) Default = *true*. Nach der Installation wird sofort eine Imagesicherung der Installation durchgeführt.
- **setup_after_install**
Hier können ein oder mehrere Produkte angegeben werden, welche auf *setup* gestellt werden und somit nach Abschluß der Betriebssysteminstallation installiert werden. Dabei werden auch die Abhängigkeiten dieser Produkte mit aufgelöst.
- **wget_and_execute:**
Url (`http`) einer Datei welche am Ende der Installation geholt und ausgeführt wird. (Default = "")
- **release:**
Welches Release von Ubuntu soll installiert werden. (Default="trusty")
- **install_opsi-client-agent:**
Installiere den Linux opsi-client-agent (Kofinanzierungsprojekt: Sie benötigen eine Aktivierung durch die `/etc/opsi/modules`) . (Default=*false*)

Netboot Produkte zum Backup und Restore

- **opsi-local-image-backup**
Dieses Produkt sichert das aktuell auf Partition 1 installierte Betriebssystem in einer Imagedatei auf Partition 4. Als Image Namen wird der im Property gesetzte Name verwendet. Wenn dieser leer ist, so wird der Name des opsi Netbootproduktes verwendet, welcher aktuell auf *installed* steht (z.B. `opsi-local-image-winxp`). Dieser Name wird beim Produkt `opsi-local-image-restore` als Produktproperty *imagefile* gesetzt, so daß ein nachfolgender Aufruf von `opsi-local-image-restore` per default genau dieses Image wiederherstellt. Dieser Name wird beim Produkt `opsi-local-image-restore` dem Produktproperty *imagefiles_list* hinzugefügt. Damit enthält dieses Property die Liste aller verfügbaren Images. Weiterhin werden (für die Windows-Produkte) die aktuellen opsi-Produktstände zusammen mit dem Image gesichert, damit auch diese im Rahmen des restores wiederhergestellt werden können. Als Sicherungsmethode wird `partclone` verwendet.
Properties:
 - **askbeforeinst:**
Soll das Starten der Installation am Client bestätigt werden müssen? (Default=*false*)
 - **free_on_backup:**
Dies ist ein read-only property welches aktuelle Informationen zur Backuppartition anzeigt: `device, size, used, remaining, use in %, mount point`
 - **imagefile**
Name der zu erstellenden Imagedatei (default = leer = der Name des aktuell installierten opsi-local-image Produktes wird verwendet). Der Name darf Leerzeichen aber keine Umlaute enthalten. (Im Fall von Leerzeichen werden diese intern als Unterstich behandelt. D.h. *mein image = mein_image*.)
 - **setup_after_install**
Hier können ein oder mehrere Produkte angegeben werden, welche auf *setup* gestellt werden und somit nach Abschluß installiert werden. Dabei werden auch die Abhängigkeiten dieser Produkte mit aufgelöst.
- **opsi-local-image-restore**
Dieses Produkt spielt das im Produktproperty *imagefile* angegebene Image wieder auf der Partition 1 ein und sorgt dafür, dass es gebootet werden kann. Weiterhin werden (für die Windows-Produkte) die für das Image gesicherten opsi-Produktstände zusammen mit dem Image wiederhergestellt.
Properties:
 - **askbeforeinst:**
Soll das Starten der Installation am Client bestätigt werden müssen? (Default=*true*)
 - **architecture:**
Architektur Auswahl, beeinflusst die Auswahl des bootimages und die Installationsarchitektur. (Default=*64bit*)

- **imagefile**
Name des Images welches restored werden soll. Der Wert dieses Properties wird automatisch durch das letzte Backup gesetzt. Die Liste der verfügbaren Images findet sich im Property `imagefiles_list`.
 - **imagefiles_list**
Liste der verfügbaren Images. Wird durch das Backup Produkt gepflegt.
 - **update_and_backup**
(*true/false*) Default = *false*. Bei *true* wird nach dem restore dafür gesorgt, dass alle Localboot Produkte die auf dem Server in einer abweichenden Version vorhanden sind auf **setup** gestellt werden und das Produkt `opsi-local-image-backup-starter` auf **once** gesetzt wird. Dies führt dazu, dass alle vorhandenen Updates eingespielt und nach den Updates automatisch wieder eine Sicherung durchgeführt wird.
 - **setup_after_restore**
Hier können ein oder mehrere opsi-Produkte angegeben werden, welche nach dem Abschluß des Restores auf *setup* gesetzt werden und somit nach dem Reboot automatisch ausgeführt werden. Der Default ist das Produkt *windomain* zur erneuten Aufnahme des restorten Clients in die Windows Domain.
- **opsi-local-image-delete**
Dieses Produkt löscht das im Produktproperty *imagefile* angegebene Image von der Backup Partition
- **imagefile**
Name des zu löschenden Images (default = leer = Fehler)

Localboot Produkt zur Ablaufsteuerung

- **opsi-local-image-backup-starter**
Dieses Localbootprodukt stellt das Netbootprodukt `opsi-local-image-backup` auf *setup* und rebootet den Rechner dann. Dieses Produkt hat eine sehr niedrige Priorität von -98. Dies bedeutet das alle *normalen* Localboot Produkte vorher abgearbeitet werden. Dies kann zu folgendem genutzt werden:
Werden für einen Rechner folgende Produkte auf *setup* gestellt:
- Das Produkt `opsi-local-image-restore`
- Alle Localboot Produkte die nicht aktuell sind
- Das Produkt `opsi-local-image-backup-starter`

so führt dies zu folgendem Ablauf:

1. Restore des Images
2. Update des restoreten Betriebssystems (Alle nicht aktuellen Produkte werden aktualisiert)
3. Backup des upgedateten Betriebssystems

9.7.8 Erweiterte opsi Service Methoden

Im Rahmen dieser Erweiterung können die Rechner eines Schulungsraums in einer opsi-client Gruppe zusammengefasst werden. Um möglichst komfortable Aktionen für alle Rechner eines Schulungsraums auszulösen sind folgende Erweiterungen der opsi-service Methoden vorgesehen:

- **setProductActionRequestForHostGroup**
Parameter: `hostGroupId`, `productId`, `actionRequest`
Ermöglicht für alle Mitglieder einer Gruppe (z.B. Rechner eines Schulungsraums) eine bestimmte Aktion zu starten (z.B. Restore des Images).

- **setProductPropertyForHostGroup**
Parameter: `productId` `propertyId` `propertyValue` `hostGroupId`
Ermöglicht für alle Mitglieder einer Gruppe (z.B. Rechner eines Schulungsraums) für ein bestimmtes Produkt einen Propertywert zusetzen (z.B. welches Image restored werden soll).
- **getPossibleImagefileValuesForHostGroup**
Parameter: `groupId`
Liefert die Liste der Imagefile Namen, welche das `opsi-local-image-backup` auf allen Mitgliedern der Gruppe angelegt hat. Wenn ein bestimmtes Image (z.B. `opsi-local-image-winxp`) auf einem oder mehreren Rechnern nicht vorhanden ist, so ist es nicht Bestandteil der Rückgabeliste.

Diese Methoden werden zu einem späteren Zeitpunkt in die Standardpakete von opsi integriert. Bis dahin steht Ihnen in der Auslieferung eine Datei `40_groupActions.conf` zur Verfügung die Sie bitte mit `root` Rechten nach `/etc/opsi/backendManager/extend.d` kopieren. Führen Sie danach aus: `opsi-setup --set-rights /etc/opsi`.

9.7.9 Backuppartition

Die Backuppartition ist (bei MBR BIOS und ohne Datenpartition) die dritte Partition der System-Festplatte. Bei Systemen mit mehr als einer Platte wird die Systemfestplatte bestimmt durch das `opsi-local-image-prepare` Property `multi_disk_mode`.

Bei Systemen mit mehr als einer Platte kann sich die Backuppartition abhängig vom `opsi-local-image-prepare` Property `backup_partition_on_same_disk` auch als erste Partition einer anderen Platte befinden.

Dort finden sich:

- Die Datei `master.log` mit Informationen über alle durchgeführten Image Operationen. Diese Logdatei wird in die bootimage logs übernommen.
- Die Image Verzeichnisse
Die Imageverzeichnisse haben den selben Namen wie das Image und enthalten neben dem Image noch die Metadaten des Images.
Zur Orientierung hier beispielhaft die Größen von unterschiedlichen Imageverzeichnissen mit OS und Standardsoftware (Libreoffice, Adobereader, firefox, thunderbird, javavm, flashplayer):
- `opsi-local-image-ubuntu`: 3.6G
- `opsi-local-image-winxp`: 6.4G
- `opsi-local-image-win7`: 9.4G
- `opsi-local-image-win7-x64`: 13G

9.7.10 Capture Images (WIM) erstellen und verteilen

Capture Images (WIM) Einführung

Microsoft hat mit NT6 (also ab Vista) zur Installation ein neues Imageformat, das **Windows Imaging Format (WIM)** eingeführt. Ein WIM Image ist kein Platten- oder Partitionsimage sondern mehr ein Dateien und Metadaten Archiv. Eine WIM Datei kann mehrere Images enthalten. Die normale Installation eines NT6 Rechners basiert darauf, dass die `setup.exe` ein Image aus der Datei `install.wim` auspackt und dieses danach konfiguriert und mit zusätzlichen Treibern versieht.

Von einem existierender Rechner kann das Windows inclusive installierter Software, Hotfixes und Konfigurationen ausgelesen und in Form eines WIM abgespeichert werden. Ein solches WIM kann dann wieder die Basis für neue Installationen sein.

Capture Images (WIM) Komponenten

Für die Erstellung eines Capture Images im Wim Format benötigen Sie ab den Produktversionen 4.1 nur noch das Produkt:

- opsi-local-image-wim-capture

Die vorherigen Produkte:

- opsi-local-image-capture
- opsi-local-image-sysprep

Sind damit obsolet und können gelöscht werden.

Ergänzend gibt es die *Zielprodukte* welche dafür gedacht sind, das gecapturte Image aufzunehmen:

- opsi-local-image-win7-capture
- opsi-local-image-win7-x64-capture
- opsi-local-image-win81-capture
- opsi-local-image-win81-x64-capture
- opsi-local-image-win10-capture
- opsi-local-image-win10-x64-capture

Capture Images (WIM) Abläufe

Die Abläufe und Einstellungen beim Produkt opsi-local-image-wim-capture entsprechen denen des Produktes opsi-wim-capture welches hier: Abschnitt 9.4 beschrieben ist. Die Properties von opsi-wim-capture sind hier: Abschnitt 9.4.6 beschrieben.

Der wesentliche Unterschied zwischen den beiden Produkten ist:

opsi-local-image-wim-capture verwendet zur Sicherung und Wiederherstellung der zu capturten Partition den Mechanismus von opsi-local-image-backup/opsi-local-image-restore. Für den selben Zweck verwendet opsi-wim-capture das Produkt opsi-clonezilla



Wichtig

opsi-local-image-wim-capture schlägt fehl, wenn Sie Ihr System mit einer Datenpartition angelegt haben. Installieren Sie in diesem Fall den Rechner neu mit dem opsi-local-image-prepare Property data_partition_size=0.

9.7.11 Windows Installation von einem Targetprodukt aus

(Ausrollen des gecapturten Images)

Wiederherstellung der opsi Metadaten zu installierten Produkten

Das Problem:

Wenn Sie ein Windows mit opsi neu installieren, z.B. aus win7-x64, dann werden bei der Installation des opsi-client-agent alle Localboot-Produkte, welche bei diesem Rechner vorher auf **installed** standen, automatisch auf setup gestellt und damit später erneut installiert.

Dies kann beim Ausrollen eines *gecapturten* Images nicht ganz genauso durchgeführt werden.

Im Image befindet sich das Backup der opsi-Daten, das dort während des capture Vorgangs abgelegt wurde. Dieses wird bei der Installation des opsi-client-agent entdeckt, und wieder in den opsi-server eingespielt. Damit stehen die

Produkte, die in dem *gecapturten* Image installiert waren, jetzt für den frisch installierten Rechner auf `installed`. Würden jetzt alle Produkte, welche auf `installed` stehen auf `setup` gesetzt, würde dies dazu führen, dass alle schon im Image installierten Produkte nochmal installiert werden. Dies ist nicht erwünscht.

Bei der Wiederherstellung der opsi Metadaten zu installierten Produkten gibt es ab opsi 4.0.7 zwei Varianten:

- Variante 1:
Zurückspielen der Metadaten und Beibehaltung von *setup*-Actionrequests.
Produkte die auf *installed* stehen werden **nicht** auf *setup* gestellt.
Dies ist der Default und das Verhalten vor opsi 4.0.7
- Variante 2:
Zurückspielen der Metadaten. Produkte die auf *installed* stehen werden auf *setup* gestellt ausser denen welche in den restorten Metadaten enthalten waren.

Variante 1

Beim Ausrollen eines *gecapturten* Images werden nach der Installation des Images nur die Produkte automatisch installiert, welche schon vor dem Beginn der Betriebssystem-Installation auf `setup` standen. Dies kann durch Ihren Eingriff oder das Property `setup_after_install` erfolgt sein. Daher werden in diesem Fall auch nur die Produkte installiert, welche vor der Installation des Betriebssystems auf `setup` standen.

Dies ist der Default und das Verhalten vor opsi 4.0.7

Variante 2

Die Variante 2 verhält sich vom Ergebnis ähnlich wie es bei Installationen aus nicht *gecapturten* Images der Fall ist:

* Zurückspielen der Metadaten.

* Produkte die auf *installed* stehen werden auf *setup* gestellt ausser denen welche in den restorten Metadaten enthalten waren.

Diese Verhalten steht erst ab opsi 4.0.7 zur Verfügung und ist nicht der Default. Variante 2 ist durch Erweiterungen am opsi-script möglich geworden und ist Bestandteil des opsi-client-agent von 4.0.7.

Um dieses Verhalten zu verwenden muss ein *config* (*Hostparameter*) gesetzt werden:

Der boolsche Konfigurationseintrag: `clientconfig.capture.switch_installed_products_to_setup`. Hat dieser Eintrag für den Client den Wert `true` dann wird Variante 2 verwendet, ansonsten Variante 1.

Über diese *Hostparameter* können dann Events Client-spezifisch aktiviert bzw. deaktiviert werden. Die *Hostparameter* können über den *opsi-configed* oder *opsi-admin* angelegt werden.

Zum Anlegen der *Hostparameter* über *opsi-admin* sind die folgenden Befehle auf dem *opsi-configserver* auszuführen:

```
opsi-admin -d method config_createBool clientconfig.capture.switch_installed_products_to_setup "capture.\
switch_installed_products_to_setup" true
```

Damit stellen Sie für **alle** Rechner *Variante 2* ein.

Zum Anlegen der *Hostparameter* über den *opsi-configed* wählen Sie dort *Serverkonfiguration* / *clientconfig* / Auf der Rechten Seite mit der rechten Maustaste: **Boolschen Konfigurationseintrag hinzufügen**.

9.7.12 Hilfsprodukt opsi-wim-info

Das Produkt `opsi-wim-info` kann verwendet werden um schnell Informationen über die in einer `install.wim` gespeicherten Images auszulesen. Diese Informationen werden dann in der Logdatei gespeichert.

Properties:

- `target_produk`
ProductId des Produktes in dem die *install.wim* gesucht wird.

9.7.13 Erstellen eines eigenen Ubuntu Proxy

Eine brauchbare Anleitung zur Erstellung eines eigenen Ubuntu Proxy finden Sie hier:

- link:http://wiki.ubuntuusers.de/Lokale_Paketquellen/Apt-Cacher-ng
- link:<http://www.gambaru.de/blog/2011/10/26/apt-cacher-ng-ein-proxy-server-fur-debian-und-ubuntu/>

9.8 opsi vhd reset

9.8.1 Vorbedingungen für die opsi Erweiterung 'opsi vhd reset

Dieses Modul ist momentan eine [kofinanzierte opsi Erweiterung](#).

Es ist gebündelt mit der Erweiterung *opsi-local-image* (siehe Abschnitt 9.7) - das heißt: die Freischaltung für *opsi-local-image* gilt auch für *opsi-vhd-reset*.

Es sind eine Reihe von Vorbedingungen nötig, um dieses Modul einsetzen zu können. Das bedeutet, dass Sie zum Einsatz eine Freischaltdatei benötigen. Diese Freischaltung erhalten Sie, wenn Sie die Erweiterung kaufen. Zu Evaluierungszwecken stellen wir Ihnen auch eine zeitlich befristete Freischaltung kostenlos zur Verfügung (→ mail an info@uib.de).

Weitere Details hierzu finden Sie in Abschnitt 9.1.

Technische Voraussetzungen sind opsi \geq 4.0.7 mit den Paketständen:

Tabelle 18: Benötigte Pakete

opsi-Paket	Version
opsi-winst	\geq 4.12.0.13

9.8.2 Einführung

Um in Schulungsräumen Rechner innerhalb von kurzer Zeit wie z.B. in einer Pause zwischen zwei Kursen wieder in einen definierten Zustand zu versetzen bedarf es besonderer Mittel. Mit *opsi-local-image* bietet opsi hier bereits etwas an, das nun ergänzt wird durch eine neue Methode welche spezifische Vor- und Nachteile hat.

1. Initiale Windows 10 Installation in einen VHD Container
2. *Versiegelung* der initialen Installation durch eine *child* VHD
3. Schnelle Wiederherstellung durch Austausch der *child* VHD.
4. Upgrade der initialen Installation durch einen merge der *child* VHD.
5. Das Verfahren arbeitet mit den aus der Virtualisierung bekannten snapshot Techniken ohne selbst eine Virtualisierung zu benötigen.

9.8.3 Abläufe

Initiale Installation

Über das Produkt *opsi-vhd-win10-x64* ein Windows 10 in einen VHD-Container installiert.

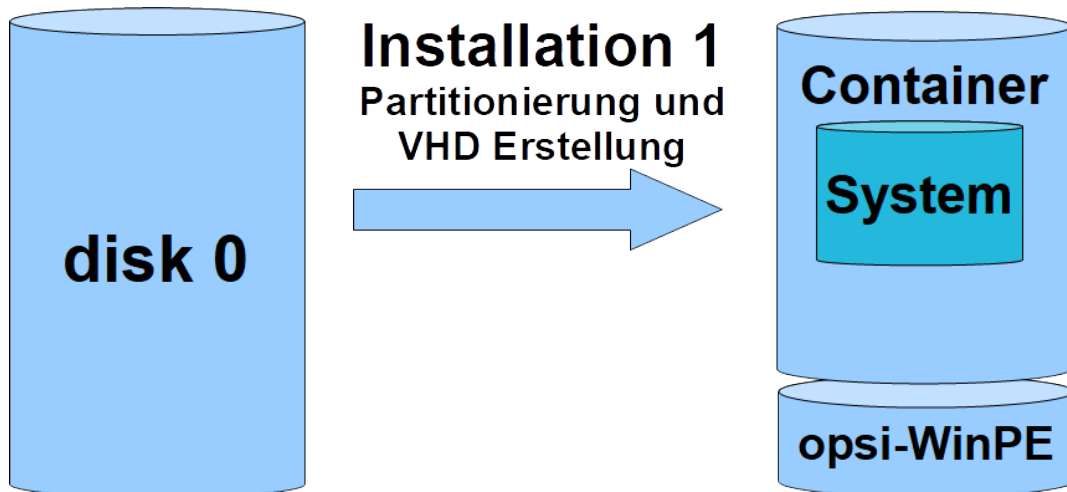


Abbildung 96: Schema: Initiale Installation 1: Erstellen der VHD

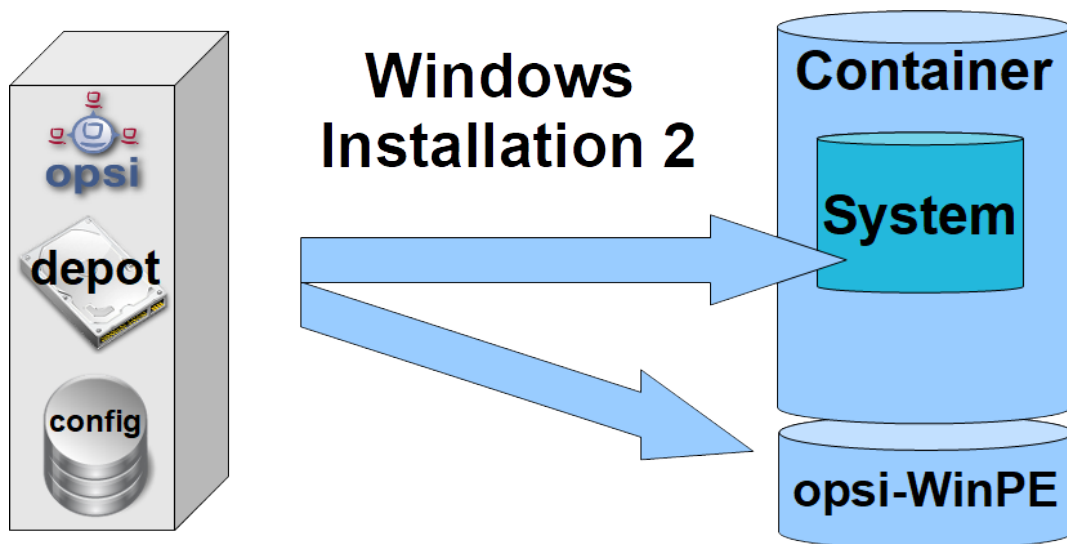


Abbildung 97: Schema: Initiale Installation 2: Windows Installation

Anschließend können in dieses Windows die gewünschten Applikationen installiert werden.

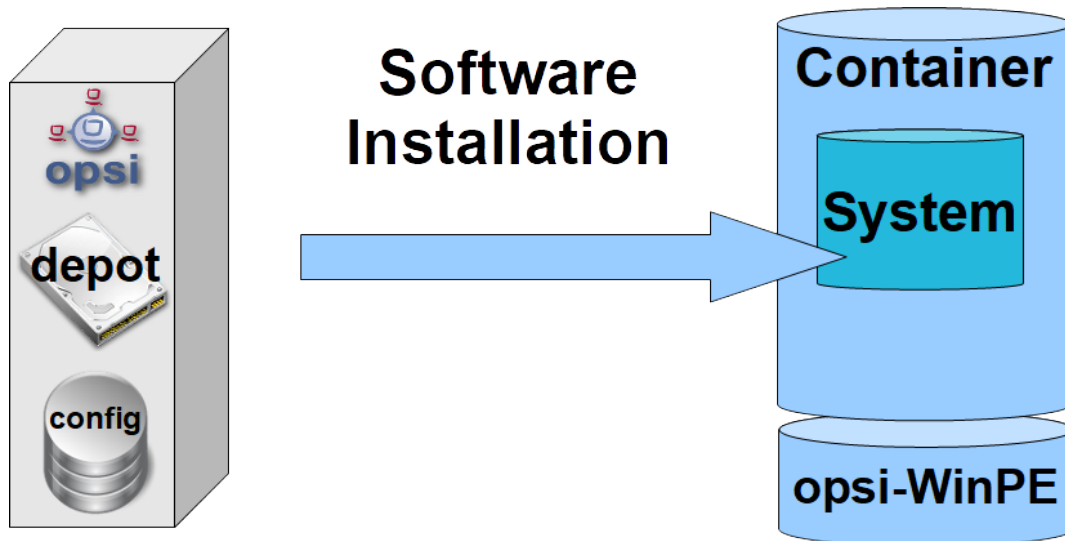


Abbildung 98: Schema: Initiale Installation 3: Software Installation

Durch einen Aufruf des opsi Produktes *opsi-vhd-control* werden zunächst die aktuellen opsi Meta Daten zu diesem Client (welches Produkt ist in welcher Version installiert) in der initialen Installation abgelegt. Anschließend wird für die weiteren Vorgänge das Windows PE aktiviert und gebootet. Das Produkt *opsi-vhd-control* hat eine sehr niedrige Priorität (-97) und kommt daher erst nach der Installation von Anwendungssoftware an die Reihe. Dies hat zur Folge, dass das Produkt *opsi-vhd-control* schon zusammen mit der Anwendungssoftware auf *setup* gestellt werden kann.

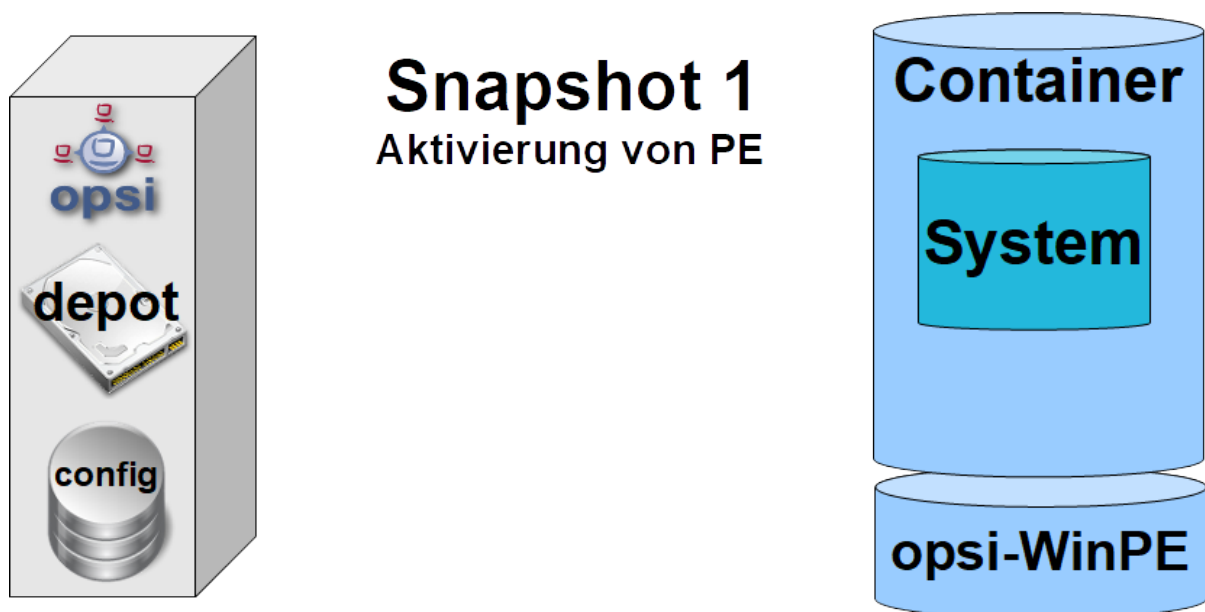


Abbildung 99: Schema: Initiale Installation 4: Aktivierung der PE Partition

Vom Windows PE aus wird durch anlegen einer Child VHD die initiale Installation gegen Veränderungen geschützt.

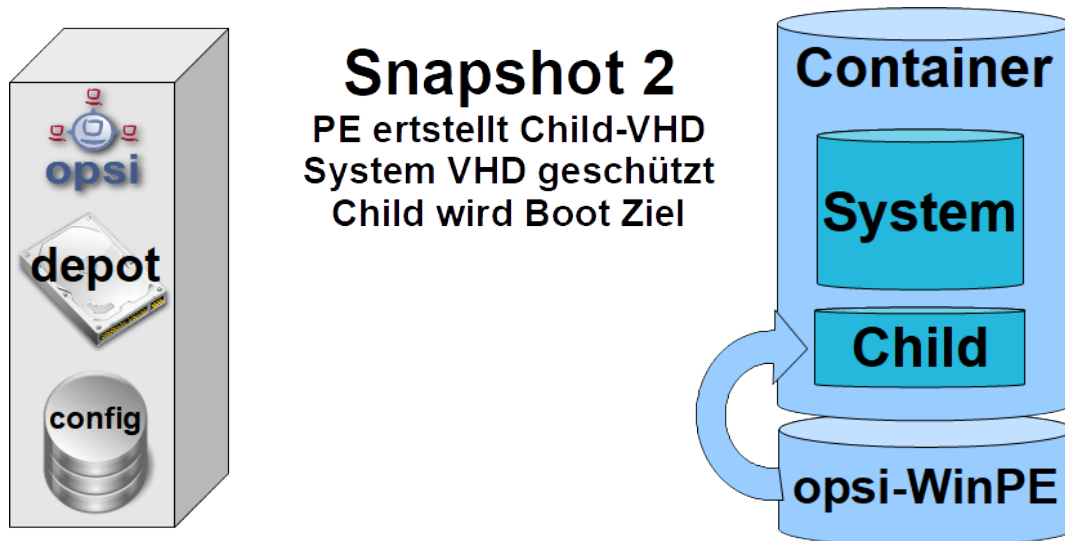
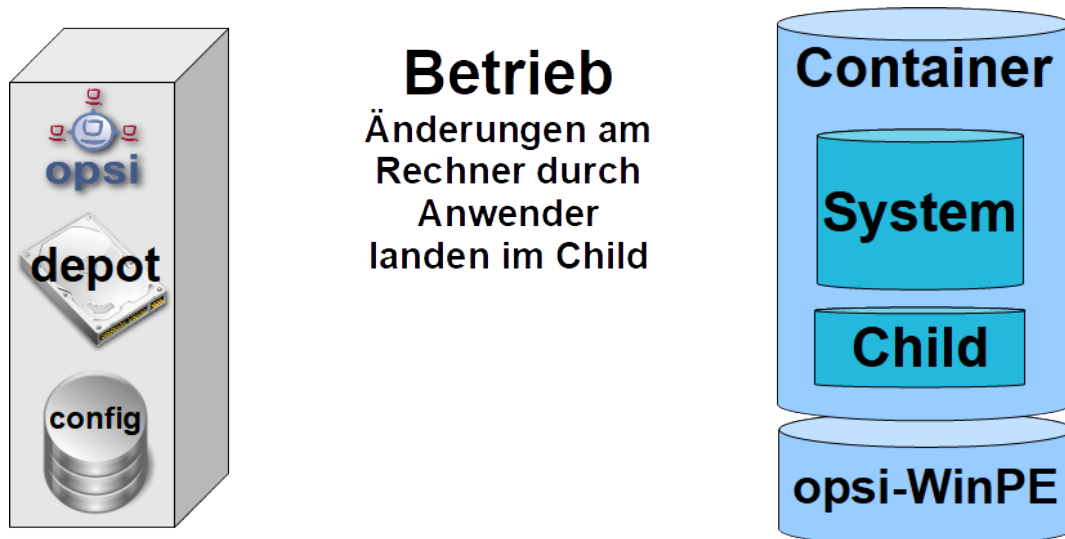


Abbildung 100: Schema: opsi-vhd-control: Versiegelung der initialen Installation

Änderungen landen ab jetzt in der *child* VHD.

Abbildung 101: Schema: Arbeiten mit dem *versiegelten* System

Schnelle Wiederherstellung

Über das opsi Produkt *opsi-vhd-control* kann die Initiale Installation wieder hergestellt werden. Zunächst werden die gespeicherten opsi Meta Daten aus dem System wiederhergestellt. Dann wird für das Child VHD handling wieder in das Windows PE gebootet.

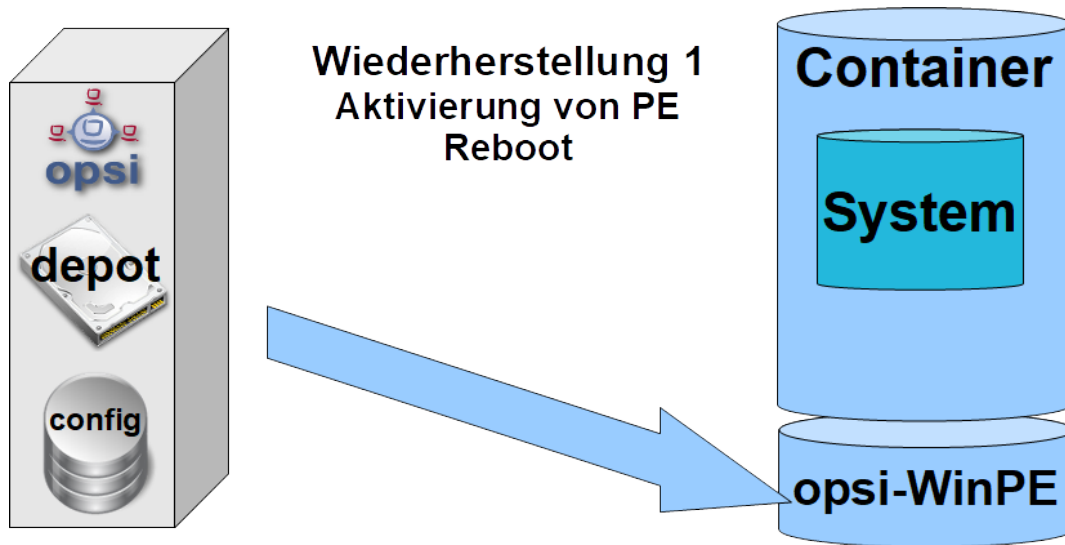


Abbildung 102: Schema: opsi-vhd-control: Wiederherstellung der initialen Installation 1

Vom Windows PE aus wird die *child* VHD mit den Veränderungen gelöscht und gegen eine neue, leere *child* VHD ausgetauscht.

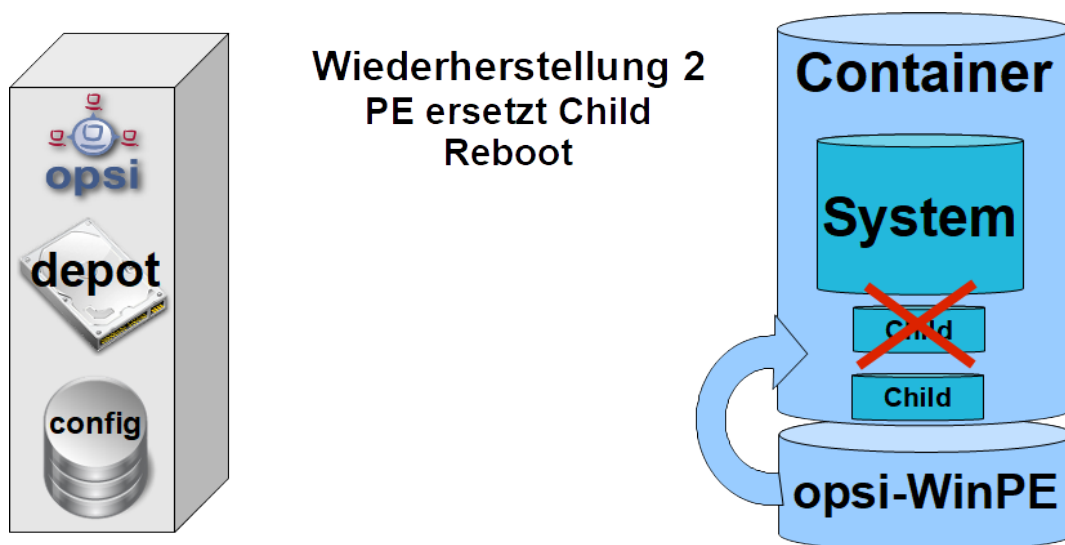


Abbildung 103: Schema: opsi-vhd-control: Wiederherstellung der initialen Installation 2

Update eines Images mit opsi-vhd-auto-upgrade

Für ein Update der initialen Installation mit Patches und Softwareupdates, kann wie folgt vorgegangen werden:

- Wiederherstellung der initialen Installation (wie oben beschrieben)
- Einspielen der Updates
- Integration der Updates in die initiale Installation und Neuversiegelung durch start von *opsi-vhd-control* mit dem Property *upgrade=true*
- Dies startet auch das Ablegen der neuen opsi Meta Daten im System

Diese Vorgänge werden automatisiert durchgeführt durch das Produkt `opsi-vhd-control` mit dem Property `upgrade=true`.

9.8.4 Die opsi-vhd Produkte

Die Erweiterung `opsi vhd reset` besteht aus folgenden Produkten

Das Netbootprodukt zur initialen Installation

- `opsi-vhd-win10-x64`

Das Localboot Produkt zur Steuerung der Erstellung, des Austausches und des Merge der Child-VHD's.

- `opsi-vhd-control`

Das Localboot Produkt zum Vollautomatischen Update der Parent VHD.

- `opsi-vhd-auto-upgrade`

UEFI Kompatibilität

Die opsi-vhd Produkte sind UEFI kompatibel.

Das opsi Netboot Produkt `opsi-vhd-win10-x64` und seine Properties

Diese Netbootprodukt gleicht vom Aufbau her den normalen Netbootprodukten (4.1.0) zur Windows Installation und muß entsprechend befüllt werden wie dies im *Getting-started* Handbuch beschrieben ist.

Auch die Properties sind weitgehend die selben.

Folgende Properties sind speziell für dieses Produkt:

- `windows_vhd_size`
Dieses Property gibt die Größe der Basis VHD absolut oder in Prozent der Festplattengröße abzüglich der WinPE Partition an. Der Defaultwert von 100% wird automatisch auf 80 % gekürzt, um Platz für die child VHD zu lassen. Wird (absolut oder relativ) ein Wert angegeben der über 80% landet, so wird dieser auch auf 80% vermindert. Dieses Property ersetzt das Standard Property `windows_partition_size` (Default = 100%)
- `installto:`
Der Wert ist `vhd` und soll und kann auch nicht geändert werden

Folgende Properties fehlen bei diesem Produkt:

- `windows_partition_size`, `windows_partition_label`
Siehe oben, Das Label der Partition in welcher die VHD's liegen ist `CONTAINER`
- `data_partition_size`, `data_partition_letter`, `data_partition_create`, `data_partition_preserve`
Die Verwaltung einer Data-Partition ist bisher bei opsi-vhd nicht vorgesehen.
- `boot_partition_size`, `boot_partition_letter`, `boot_partition_label`
Die Verwaltung einer Boot-Partition ist bisher bei opsi-vhd nicht vorgesehen.
- `pre_format_system_partitions`, `preserve_winpe_partition`
Bei opsi-vhd stehen diese beiden Werte fest auf `true`.

Das opsi Localboot Produkt *opsi-vhd-control* und seine Properties

Das Produkt *opsi-vhd-control* hat eine sehr niedrige Priorität (-96).

- **disabled**
Diese Property dient zu Debug Zwecken.
Wenn *true* führt das Produkt keine Aktionen aus.
Default = *false*
- **upgrade**
Wenn *true*: Merge die in der Child CHD gesammelten Änderungen in die Haupt VHD. Danach tausche die child VHD gegen eine leere child VHD aus.
Wenn *false*: Tausche die child VHD gegen eine leere child VHD aus.
Am Ende eines erfolgreichen *upgrade* Laufs wird dieses Property automatisch auf *false* zurückgestellt.
Default = *false*
- **stop_on_no_network_in_pe**
Diese Property dient zu Debug Zwecken.
Wenn *true*: Breche mit einer Fehlermeldung ab, damit untersucht werden kann, warum keine Netzwerkverbindung aufgebaut werden konnte. Default = *false*

Das opsi Localboot Produkt *opsi-vhd-auto-upgrade* und seine Properties

Das Produkt *opsi-vhd-auto-upgrade* hat eine sehr niedrige Priorität (-97), welche noch geringer ist als die von *opsi-vhd-control*.

- **disabled**
Diese Property dient zu Debug Zwecken.
Wenn *true* führt das Produkt keine Aktionen aus.
Default = *false*
- **rebootflag**
Bitte während des Laufes nicht ändern. Sollte vor dem Start 0 sein.
- **stop_after_step**
Diese Property dient zu Debug Zwecken.
Wenn nicht 0 dann anzahl der reboots nach denen gestopt wird. Default = 0

Bekannte Probleme und Einschränkungen

- Es gibt auch eine 32 Bit Version. Diese ist aufgrund eines Problems beim Diskpart merge Befehls in den 32 Windows PE Versionen nur eingeschränkt verwendbar.
- Theoretisch wäre auch eine Implementierung für Windows 8.1 bzw Windows 7 Enterprise möglich. Diese werden wir aber nur auf Bedarf anfertigen.
- Es gibt Hinweise darauf, das ein Windows 10 Release Upgrade einer Installation in einer VHD fehlschlägt. (<https://www.heise.de/newsticker/meldung/VHD-Boot-Windows-Update-demoliert-Aktivierung-3806023.html>)

9.9 opsi-Lizenzmanagement

9.9.1 Vorbedingungen für die opsi-Lizenzmanagement-Erweiterung

Dieses Modul ist momentan eine kofinanzierte opsi Erweiterung. Das bedeutet, die Verwendung ist nicht kostenlos. Weitere Details hierzu finden Sie in Abschnitt 9.1.

9.9.2 Überblick

Funktion und Features

Das opsi-Lizenzmanagement-Modul ist darauf ausgerichtet, die aufwändige und komplexe Verwaltung von Lizenzen für die diversen nicht-freien Softwareprodukte, die auf mit opsi verwalteten Clients eingesetzt werden, zu vereinheitlichen und zu vereinfachen.

Die wesentlichen Features sind:

- Handhabung der Lizenzverwaltung innerhalb der gleichen Oberfläche wie Softwareverteilung und Betriebssysteminstallation, d.h. im opsi-Konfigurationseditor.
- Automatische Bereitstellung, Zuteilung und Reservierung der Lizenzkeys.
- Verfügbarkeit verschiedener Lizenzmodelle
 - Standard-Einzellizenz (1 Installation, eventuell ausgewiesen durch einen Lizenzkey, für einen beliebigen Rechner)
 - Volume-Lizenz (1 Lizenzkey - eine bestimmte Zahl von Installationen) oder Campus-Lizenz (1 Schlüssel - unbegrenzte Zahl von Installationen)
 - PC-gebundene Lizenz,
 - Concurrent License (Lizenzserver-vermittelt)
- Freigabe der Lizenzkeys bei der Deinstallation von Software.
- Manuelle Bearbeitung der Lizenzzuordnungen z.B. für Lizenzen von Software, die nicht mit opsi verteilt werden.
- Report-Funktion auch für nicht mit opsi verwaltete Lizenznutzungen auf der Basis der Software-Inventarisierung und Abgleich mit den opsi-basierten Nutzungen.

Übersicht Datenbankmodell

In der Welt der nicht freien Software ist Lizenzverwaltung stets ein komplexes Thema. Es abzubilden verwendet opsi auch ein relativ komplexes Datenbankmodell. Zur Übersicht sind die beteiligten Tabellen in einem Diagramm dargestellt. Die Bedeutung der verschiedenen Relationen sollte sich nach und nach aus den folgenden Erläuterungen ergeben.

Die blaue Linie im Diagramm markiert die Abgrenzung von Tabellen, deren Inhalte automatisiert aus dem Software-Audit kommen, und Tabellen, deren Daten eigens zur Lizenzverwaltung erhoben oder zugeordnet werden. Das Diagramm zeigt dabei bereits die Bedeutung der Konstruktion der Lizenzpools (license pools), da die Tabelle der Lizenzpools die Verknüpfung zwischen den beiden Datensphären herstellt.

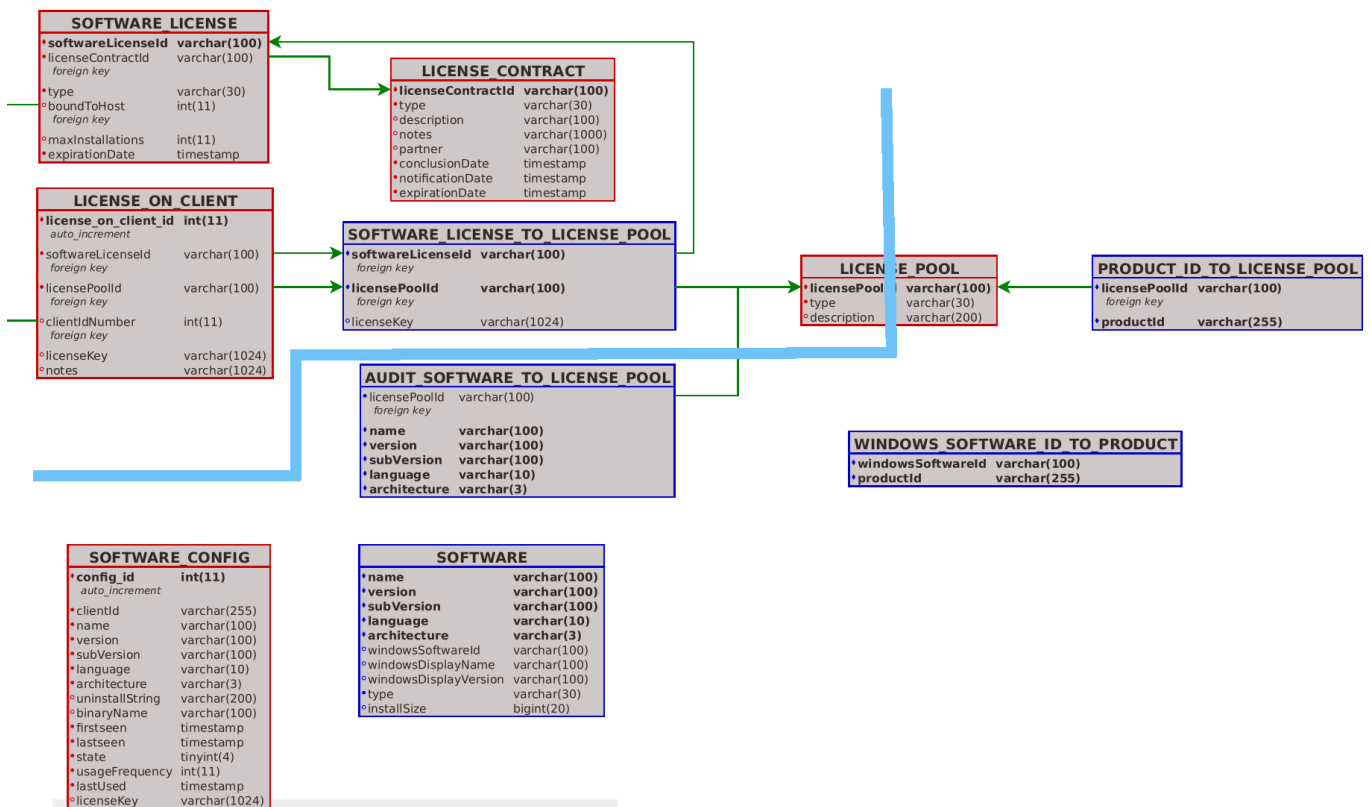


Abbildung 104: Datenbanktabellen relevant für das Lizenzmanagement

Aufruf der Lizenzmanagement-Funktionen im *opsi-configed*

Der *opsi-configed* verfügt für das Lizenzmanagement über ein gesondertes Fenster.

Es ist über die Schaltfläche "Lizenzen" im Hauptfenster des Konfigurationseditors erreichbar, sofern das Lizenzmanagement-Modul in der aktuellen opsi-Konfiguration aktiv ist (vgl. den Eintrag für "license management" im Hauptmenü unter */Hilfe/Module*).

Bei nicht aktiviertem Lizenzmanagement wird lediglich ein Hinweis angezeigt.



Abbildung 105: opsi-configed: Leiste mit Schaltfläche "Lizenzen" (rechts)

Das Modul opsi-Lizenzmanagement ist als ein kofinanziertes opsi-Erweiterungsprojekt realisiert.

9.9.3 Lizenzpools

Was ist ein Lizenzpool?

Für jede Art von benötigten Lizenzen ist im opsi-Lizenzmanagement ein *Lizenzpool* (*license pool*) einzurichten.

Der Lizenzpool ist dabei ein Konstrukt, das die gedankliche Zusammenfassung aller Lizenzen beschreibt, die für eine bestimmte Art von installierter oder zu installierender Software vorrätig sind.

Dieses Konstrukt steht im Mittelpunkt aller Aktivitäten im opsi-Lizenzmanagement.

Demgemäß ist die erste Tab-Seite des Lizenzmanagement-Fensters im *opsi-configed* der Administration der Lizenzpools gewidmet.

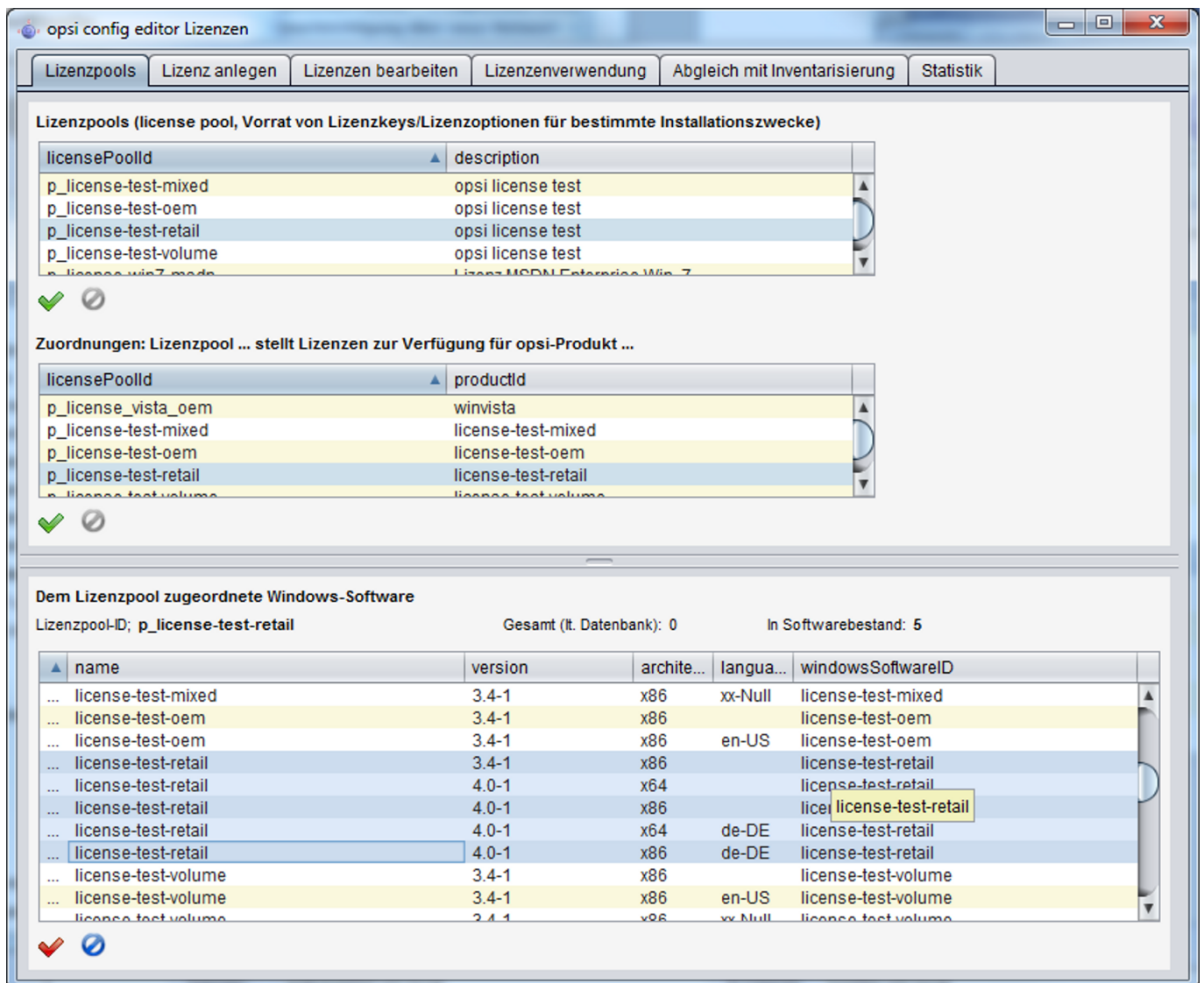


Abbildung 106: Lizenzmanagement:Tab Lizenzpools

Verwaltung von Lizenzpools

Auf der Lizenzpool-Seite befindet sich im oberen Bereich die Tabelle der verfügbaren Lizenzpools.

Hier ist das Feld *description* editierbar.

Weitere Bearbeitungsfunktionen erschließen sich über das Kontextmenü, am wichtigsten: das Erzeugen eines neuen Lizenzpools.

Nach dem Einfügen einer neuen Zeile in die Tabelle muss eine (eindeutige) *licensePoolId* in das entsprechende Feld eingetragen werden, z.B. *softprod_pool*. Bitte dabei keine Umlaute etc. verwenden. Eingegebene Großbuchstaben werden beim Speichern automatisch in Kleinbuchstaben konvertiert.

Die ID kann nur bis zum ersten Speichern noch bearbeitet werden. Als Schlüssel des Datensatzes ist sie danach unveränderlich.

In der Maske aktiviert jeder Bearbeitungsvorgang die Statusanzeige in der Art, dass die Farbe des O.K.-Buttons (Häkchen) von grün nach rot wechselt und auch der Cancel-Button seinen aktiven Modus annimmt. Durch Betätigen des betreffenden Buttons (oder mittels Kontextmenü) kann die Veränderung dann permanent gemacht (gespeichert) bzw. widerrufen werden.

Lizenzpools und opsi-Produkte

Im Standardfall gehört zu einem opsi-Produkt, das ein lizenzpflichtiges Software-Produkt installiert (z.B. den *Acrobat Writer*), genau ein Lizenzpool, aus dem die benötigten Lizenzen geschöpft werden.

Durchaus gängig ist, dass mehrere Produkte zu einem Lizenzpool gehören, da es sich um Varianten des selben Produktes handelt (z.B. die Produkte *win10-x64* und *opsi-local-image-win10-x64* nutzen den selben Lizenzpool *p_win10-x64*).

Weniger übersichtlich (und möglichst zu vermeiden) ist die Situation, wenn ein opsi-Produkt mehrere lizenzpflichtige Software-Produkte installiert, etwa wenn zu einem Paket "Designerprogramme" sowohl *Adobe Photoshop* wie auch *Acrobat Writer* gehören sollen.

Das opsi-Produkt muss dann Lizenzen aus mehreren Pools anfordern. Da es gleichzeitig auch weitere opsi-Produkte geben kann, die z.B. Lizenzen aus dem Pool für den *Acrobat Writer* benötigen, kann auch die umgekehrte Beziehung, vom Lizenzpool zum opsi-Produkt, gelegentlich mehrdeutig sein. Derartige Mehrdeutigkeiten können natürlich durch eine entsprechende Policy beim Produktbau vermieden werden.

Tipp

Bei lizenzpflichtiger Software packen Sie nur eine Software in ein opsi-Produkt. Weisen Sie dieses Produkt dem entsprechenden Lizenzpool zu. (Nur dann funktioniert auch die Verwendung des Lizenzmanagements mit der 'opsi-WAN-Erweiterung'; siehe Kapitel Abschnitt [9.10](#))

Im zweiten Abschnitt der Lizenzpool-Seite wird die Tabelle aller Zuordnungen zwischen Lizenzpools und den product-Ids von opsi-Produkten dargestellt.

Wie in allen anderen Tabellen des Lizenzmanagements wird durch einen Klick auf einen Spaltentitel die Tabelle nach dem Wert in der betreffenden Spalte umsortiert;

nochmaliges Klicken ändert die Sortierungsrichtung.

Die Sortierung kann genutzt werden, um alle Zuordnungen von opsi-Produkten zu einem Lizenzpool zusammenhängend darzustellen oder umgekehrt alle einem opsi-Produkt zugeordneten Lizenzpools zu erkennen.

Über das Kontextmenü ist wieder die Funktion erreichbar, mit der eine neue Tabellenzeile, hier also eine neue Zuordnung Lizenzpool-Produkt-ID, erstellt werden kann. Zur Eingabe von Lizenzpool-ID und Produkt-ID wird bei Klick in das Tabellenfeld jeweils die Liste der verfügbaren Werte angezeigt, aus der ein Wert ausgewählt werden kann.

Lizenzpools und Windows-Software-IDs

Die dritte Tabelle der Seite "Lizenzpools" stellt in Form einer Mehrfachauswahl dar, welche IDs aus der Softwareinventarisierung dem in der ersten Tabelle markierten Lizenzpool zugeordnet sind.

Eine (Windows-Software-) ID ist ein eindeutiger Schlüsselwert, welcher im Rahmen des opsi-Software-Audits ermittelt und an den Server übertragen wird. Die Zuordnungen zu Lizenzpools können bearbeitet werden, indem die Mehrfachauswahl verändert wird (wie üblich durch Strg-Mausklick bzw. Shift-Mausklick).

Das Kontextmenü der Tabelle bietet die Option, zwischen der Anzeige nur der aktuell zugeordneten IDs oder sämtlicher im Software-Audit erfassten IDs umzuschalten.

Die Zuordnung zwischen Windows-Software-IDs und Lizenzpools dient im Lizenzmanagement vor allem dazu, nicht nur die Installationen per opsi-Setup, sondern auch die Gesamtzahl der faktischen Installationen einer Software (wie aus der Registry der Clients ausgelesen) zu kontrollieren und abzugleichen mit der Zahl der dokumentiert verfügbaren Lizenzen eines Lizenzpools (Tab "Statistik", s. unten).

9.9.4 Einrichten von Lizenzen

Das Einrichten einer Lizenz bzw. die Bereitstellung einer Lizenz in einem Lizenzpool, erfordert mehrere Schritte. Sie können, mit vorgegebenen Optionen vorstrukturiert, auf der zweiten Tab-Seite des Lizenzmanagement-Fensters (Titel "Lizenz anlegen") durchgeführt werden.

Die Seite startet mit einer (hier nicht editierbaren) Tabelle der verfügbaren Lizenzpools. Dort ist zunächst der Pool auszuwählen, für den eine Lizenz eingerichtet werden soll.

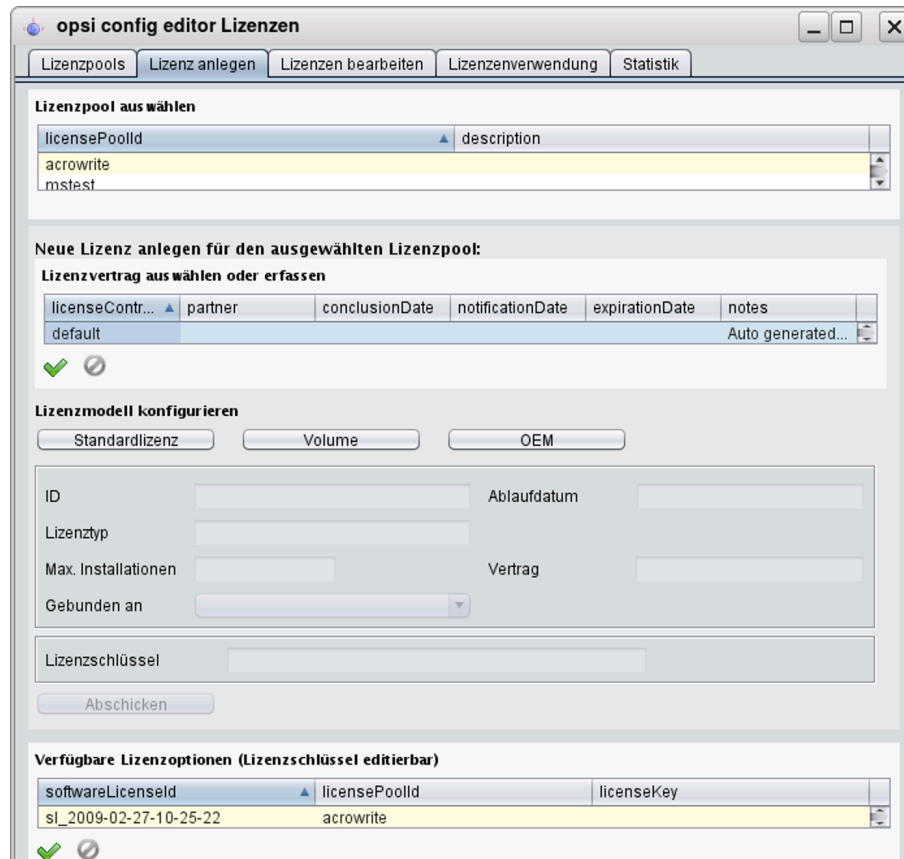


Abbildung 107: Lizenzmanagement:Tab "Lizenz anlegen"

Bevor die weitere Schritte beschrieben werden, empfiehlt es sich, einige Begrifflichkeiten zu klären:

Aspekte des Lizenzkonzepts

Unter **Lizenzierung** (*licensing*) soll die faktische Zuweisung der Erlaubnis zur Nutzung einer Software (durch Installation einer Software) verstanden werden. Sie schließt oft, aber nicht notwendig die Nutzung eines hierfür bestimmten **Lizenzschlüssels** (*license key*) ein.

Das **Lizenzierungsrecht** ist die in ihrem Geltungsumfang definierte Erlaubnis, solche Zuweisungen durchführen zu dürfen. In der opsi-Datenbank wird das Lizenzierungsrecht als *software license* bezeichnet, ein entsprechender Datensatz ist demgemäß identifiziert durch eine *softwareLicenseId*. Verschiedene Varianten der konkreten Ausgestaltung des Lizenzierungsrechts (z.B. für wie viele PCs, mit welcher Gültigkeitsdauer etc.) werden als **Lizenzmodelle** bezeichnet. Ein Lizenzierungsrecht gründet in einem **Lizenzvertrag** (*license contract*), der es im juristischen Sinn feststellt und dokumentiert.

Eine **Lizenzierungsoption** definiert die Anwendungsmöglichkeit eines Lizenzierungsrechts für einen bestimmten Lizenzpool. In opsi ist die Lizenzierungsoption festgelegt durch die Kombination einer *softwareLicenseId* und einer

licensePoolId. Zur Lizenzierungsoption gehört auch der Wert eines spezifischen Lizenzschlüssels (*licenseKey*, sofern er für eine Installation erforderlich ist).

Schließlich dokumentiert eine **Lizenznutzung** die "gezogene" Lizenzierungsoption, d.h. die erfolgte Anwendung einer Lizenzierungsoption für einen Client. Sie ist die vollzogene und berechtigte Lizenzierung einer Softwareinstallation. Beschrieben wird sie durch die Kombination *softwareLicenseId*, *licensePoolId* und dem eindeutigen Namen des betreffenden Clients, *hostId*. Der verwendete Lizenzschlüssel (*licenseKey*) wird ergänzend notiert.

Lizenzvertrag erfassen

Nach der Auswahl des Lizenzpools, für den eine Lizenzierungsoption angelegt werden soll, ist im zweiten Schritt der Lizenzvertrag zu bestimmen, auf den die Lizenzierung letztlich gründen soll. Im Seitenabschnitt "Lizenzvertrag auswählen oder erfassen" auf der Tab-Seite "Lizenz anlegen" kann ein vorhandener spezifischer Vertrag in der Tabelle ausgewählt oder ein neuer Vertrags-Datensatz angelegt werden.

In einem Lizenzvertrags-Datensatz werden wichtige Ordnungsgesichtspunkte für einen Vertrag in den Feldern (Vertrags-) *partner*, Abschlussdatum (*conclusion date*), Benachrichtigungsdatum (*notification date*) und Auslaufdatum (*expiration date*) dokumentiert. Hinzu kommt ein freies Notizfeld (*notes*), um z.B. den Aufbewahrungsort für das Realdokument eines Vertrages aufzunehmen. Die Vertrags-ID (*licenseContractId*) dient zur Identifizierung des Lizenzvertrags in der Datenbank.

Die Erfassung eines neuen Datensatzes wird über das Kontextmenü gestartet. Es werden automatisch Standard-Einträge generiert, insbesondere eine aus der aktuellen Zeit generierte Vertrags-ID und als Vertragsabschlussdatum der aktuelle Tag. Wenn die Vertragsbedingungen sich z.B. aus einem Software-Kauf implizit ergeben bzw. anderweitig dokumentiert und verfolgt werden können, können die Standard-Einträge belassen werden. Andernfalls sind hier Werte einzugeben, die eine geordnete Verfolgung des zugrundeliegenden Vertrags z.B. durch Verweis auf ein Aktenzeichen im Feld *notes* erlauben.

Die Vertrags-ID kann nur bearbeitet werden, solange der Datensatz nicht gespeichert ist.

Lizenzmodell konfigurieren

Der dritte Seitenabschnitt der Tab-Seite "Lizenz anlegen" dient dazu, die Ausgestaltung des einzurichtenden Lizenzierungsrechts festzulegen.

Es werden verschiedene Varianten angeboten:

- Standardlizenz
- Volumen-Lizenz
- OEM-Lizenz
- Concurrent-Lizenz

Jede Option ist durch einen Button repräsentiert, bei dessen Betätigung die Felder im folgenden Formularbereich vor ausgefüllt werden.

Standardlizenz soll bedeuten, dass die Lizenz zu einer Einzel-Installation der Software berechtigt und diese auf einem beliebigen PC erfolgen kann. Ein ggf. erfasster Lizenzschlüssel wird nur für eine Installation verwendet.

Eine **Volumen-Lizenz** legitimiert n Installationen, ggf. mit ein- und demselben Lizenzschlüssel. $n = 0$ soll dabei bedeuten, dass innerhalb des Netzes der Schlüssel beliebig oft zu Installationen verwendet werden darf (**Campus-Lizenz**).

Als **OEM-Lizenz** wird die Situation bezeichnet, dass eine Lizenz nur für einen, festzulegenden PC genutzt werden darf. Dies ist häufig die intendierte Lizenzart, wenn ein PC mit vorinstalliertem Betriebssystem gekauft wird.

Die **Concurrent-Lizenz** ist aus opsi-interner Sicht eine Volumenlizenz, die beliebig häufig genutzt werden darf. Mit der Auszeichnung des Lizenzmodells als Concurrent-Lizenz ist nach außen jedoch die Aussage verbunden, dass die Anzahl der faktisch in Anspruch genommenen Lizenzierungen auf andere Weise kontrolliert wird, z.B. durch einen Lizenzserver.

Wenn einer der Buttons betätigt wird, erhält auch das ID-Feld eine Vorschlagsbelegung mit einem auf der Basis von Datum und Zeit generierten String, der bearbeitet werden kann.

Je nach Lizenztyp können die anderen Felder editiert werden oder sind unveränderlich.

Das Feld "Ablaufdatum" definiert die technische Gültigkeitsgrenze des Lizenzierungsrechts (während das inhaltlich gleichbedeutende Feld *expirationDate* der Lizenzvertragstabelle Dokumentationszwecken dient). Es ist allerdings nur für einen künftigen Gebrauch vorgesehen, eine Verwendung ist derzeit nicht implementiert.

Abschicken der Daten

Der Button "Abschicken" veranlasst, dass die erfassten Daten an den opsi-Service gesendet und - sofern kein Fehler auftritt - permanent in die opsi-Datenhaltung überführt werden.

Dabei werden Datensätze für ein Lizenzierungsrecht (*software license*) basierend auf dem ausgewählten Vertrag und eine darauf bezogene Lizenzierungsoption erzeugt.

Die Liste der verfügbaren Lizenz(ierungs)optionen, die im unteren Seitenabschnitt dargestellt ist, wird automatisch neu geladen und die Markierung auf die neu erzeugte Option gesetzt.

An dieser Stelle kann, falls erforderlich, der erfasste Lizenzschlüssel korrigiert werden.

9.9.5 Lizenzierungen bearbeiten

In neunzig Prozent der Anwendungsfälle werden die Eingabe- und Editiermöglichkeiten der Tab-Seiten "Lizenzpools" und "Lizenz anlegen" genügen, um Lizenzoptionen zu erfassen und zu editieren.

Weitere Details der Lizenzkonfiguration macht die Tab-Seite "Lizenzen bearbeiten" zugänglich. Sie präsentiert die Interna der Lizenzierungsoptionen in drei Tabellen und erlaubt ggf. deren Anpassung an spezifische Erfordernisse.

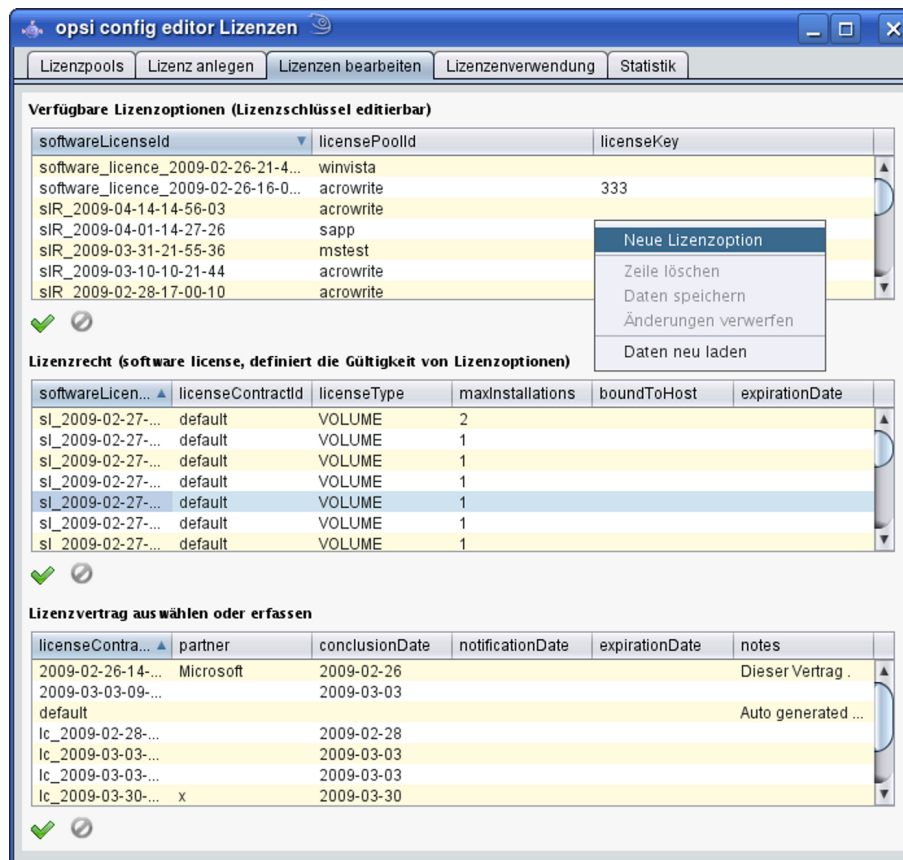


Abbildung 108: Lizenzmanagement:Tab "Lizenzierungen bearbeiten"

Im folgenden Abschnitt wird gezeigt, wie eine Lizenz mit Downgrade-Option konfiguriert werden kann, wie sie z.B. von Microsoft beim Kauf einer Windows-7-Professionallizenz angeboten wird.

Beispiel Downgrade-Option

Die Downgrade-Option bedeutet, dass anstelle der gekauften Software auch die entsprechende Vorgängerversion, z.B. Windows XP anstelle von Windows Vista, installiert werden darf. Bei diesem Microsoft-Modell darf irgendein für die Vorgängerversion vorhandener Lizenzschlüssel für eine zusätzliche Installation verwendet werden, für die er ursprünglich nicht legitimiert war.

Im opsi-Modell kann diese Konstruktion folgendermaßen abgebildet werden:

Auf der Tab-Seite "Lizenz anlegen" wird die Vista-Lizenz regulär erfasst. Das Ergebnis der Prozedur ist eine neue Lizenzierungsoption (angezeigt in der entsprechenden Tabelle am Seitenende), die auf einem gleichfalls neu angelegten Lizenzierungsrecht beruht. Letzterer ist identifizierbar durch den Wert von *softwareLicenseId*.

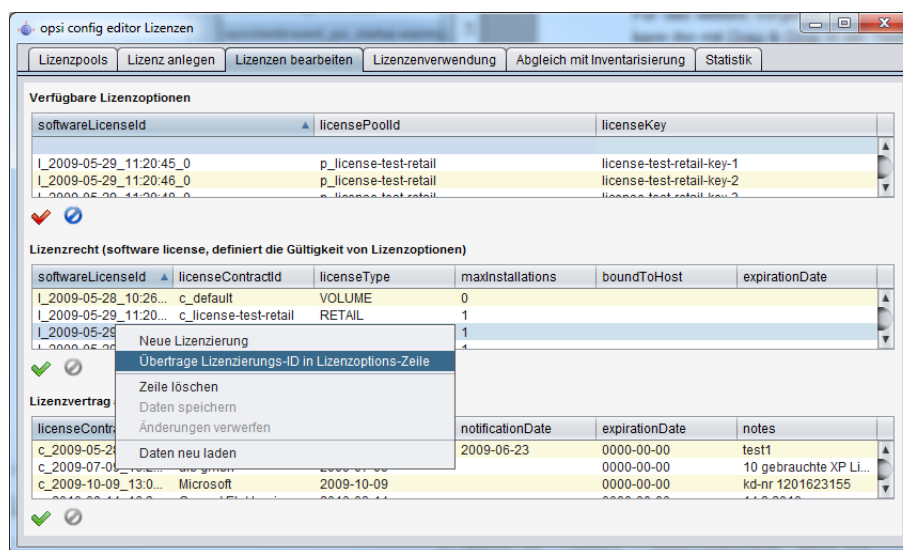


Abbildung 109: Lizenzmanagement:Lizenzmanagement:Kopieren der License-ID in die Lizenzoptionen über das Kontext-Menü

Für das weitere Vorgehen wird dieser Wert benötigt. Man kann ihn sich merken oder kann einen Editor als Zwischenablage nutzen und ihn dorthin mit *Drag & Drop* übertragen. Oder man sucht ihn auf der Tab-Seite "Lizenzen bearbeiten" in der dortigen Tabelle der Lizenzierungsrechte wieder heraus (bitte das Kontextmenü der Tabelle beachten: hier findet sich eine Spezialfunktion zum Kopieren der ID).

Der entscheidende Schritt besteht nun darin, eine Verknüpfung des gegebenen Lizenzierungsrechts mit einem zusätzlichen Lizenzpool herzustellen.

Dazu ist auf der Tab-Seite "Lizenzen bearbeiten" in der Tabelle der verfügbaren Lizenzoptionen ein neuer Datensatz anzulegen. In die betreffenden Felder des Datensatzes sind die ID des Lizenzierungsrechts, die *softwareLicenseId*, sowie die ID des zusätzlichen Lizenzpools - im Beispiel die für Windows XP - einzutragen. Für die Installation von Windows XP ist zusätzlich ein hierfür geeigneter Schlüssel, z.B. ein bei einem anderen Client bereits verwendeter, hinzuzufügen.

Nach dem Speichern sind zwei Lizenzierungsoptionen registriert, die auf das gleiche Lizenzierungsrecht verweisen! Der opsi-Service rechnet jede Anwendung einer der beiden Optionen auf die maximale Zahl von Installationen an, die das Lizenzierungsrecht einräumt. Deshalb liefert er in dem Fall einer Downgrade-Option für eine Einzel-PC-Lizenz (mit *maxInstallations* = 1) nur *entweder* für eine Installation von *Windows Vista* _oder für eine Installation von *Windows XP* einen Schlüssel.

9.9.6 Zuteilungen und Freigabe von Lizenzen

Die Anwendung einer Lizenzierungsoption für die Installation der Software auf einem Rechner führt zu einer Lizenznutzung.

Im opsi-Kontext werden Installationen skriptbasiert automatisch durchgeführt, wobei das auf den Clients abgearbeitete (Winst-) Skript Aufrufe an den zentral laufenden opsi-Service absetzt.

Im Folgenden werden die für die Lizenzverwaltung relevanten Service-Aufrufe und Skript-Befehle kurz dargestellt.

Für weitere Informationen zur Skriptsprache und zu spezifischen opsi-Kommandos s. die entsprechenden Dokumentationen, insbesondere das opsi-Winst-Handbuch.

opsi-Service-Aufrufe zur Anforderung und Freigabe einer Lizenz

Der opsi-Service-Befehl, mit dem z.B. das setup-Skript einer Betriebssystem-Installation eine Lizenzoption "ziehen" und den benötigten Lizenzkey vom Lizenzmanagement anfordern kann, lautet `getAndAssignSoftwareLicenseKey`.

Parameter sind die ID des Hosts, auf dem installiert wird und die ID des Lizenzpools, für den die Lizenz benötigt wird. Anstelle der Lizenzpool-ID kann auch eine Produkt-ID (oder eine Windows-Software-ID) als Parameter übergeben werden, falls eine entsprechende Zuordnung von Produkt bzw. Windows-Software-ID zum Lizenzpool im Lizenzmanagement registriert ist.

Analog gibt der Befehl `deleteSoftwareLicenseUsage` (wieder parametrisiert mit `hostID` und wahlweise Lizenzpool-ID), `Product-Id` oder `Windows-Software-ID` - eine Lizenznutzung frei und führt sie in den Pool der nicht verwendeten Lizenzierungsoptionen zurück.

Für die umfassende Dokumentation der opsi-Service-Befehle zum Lizenzmanagement s. unten.

Winst-Skriptbefehle für die Anforderung und Freigabe von Lizenzen

In den Winst sind die beiden client-bezogenen Befehle des Service in einen typischen Winst-Aufruf-Syntax integriert.

Ein Winst-Skript kann mit der Funktion `DemandLicenseKey` einen Schlüssel anfordern und damit die entsprechende Lizenzierungsoption *ziehen*. Die Syntaxbeschreibung ist

```
DemandLicenseKey (poolId [, productId [, windowsSoftwareId]])
```

Die Funktion gibt den Lizenzschlüssel (kann auch leer sein) als String-Wert zurück.

```
set $mykey$ = DemandLicenseKey ("pool_office2007")
```

Der Wert kann dann für die weiteren Skriptbefehle zur Installation der Software verwendet werden.

Für die Freigabe einer Lizenzoption bzw. des Schlüssels - typischerweise in einem Winst-Deinstallationskript benötigt - existiert der Befehl `FreeLicense` mit der analogen Syntax:

```
FreeLicense (poolId [, productId [, windowsSoftwareId]])
```

Die Boolesche Funktion `opsiLicenseManagementEnabled` prüft, ob das Lizenzmanagement freigeschaltet ist und kann für Skriptvariationen verwendet werden:

```
if opsiLicenseManagementEnabled
    set $mykey$ = DemandLicenseKey ("pool_office2007")
else
    set $mykey$ = getProductProperty("productkey", "")
endif
```

Die Service-Methoden können zum Beispiel über das Kommandozeilen-Werkzeug `opsi-admin` aufgerufen werden.

Mit einem * gekennzeichnete Parameter sind optional.

Lizenzverträge

```
method createLicenseContract(*licenseContractId, *partner, *conclusionDate, *notificationDate, *expirationDate, *notes)
```

Die Methode erstellt einen neuen Lizenzvertragsdatensatz mit der ID *licenseContractId*. Wird keine *licenseContractId* übergeben, wird diese automatisch generiert. Bei Angabe der *licenseContractId* eines bestehenden Vertrages wird dieser Vertrag entsprechend bearbeitet.

Die Parameter *partner* (Vertragspartner) und *notes* (Notizen zum Vertrag) sind frei wählbare Strings. *conclusionDate* (Datum des Vertragsabschlusses), *notificationDate* (Erinnerungs-Datum) und *expirationDate* (Ablauf-Datum des Vertrags) sind im Format JJJJ-MM-TT zu übergeben (z.B. 2009-05-18). Die Methode gibt die *licenseContractId* des angelegten oder bearbeiteten Vertrags zurück.

Mit den String-Funktionen `getLastServiceErrorClass` sowie `getLastServiceErrorMessage` kann auf einen Fehler reagiert werden, wenn z.B. keine freie Lizenz mehr verfügbar ist:

```
if getLastServiceErrorClass = "None"  
    comment "kein Fehler aufgetreten"  
endif
```

Die Fehlerklasse `LicenseMissingError` wird zurückgegeben, falls eine Lizenz angefordert wird, aber nicht verfügbar ist.

Die Fehlerklasse `LicenseConfigurationError` wird zurückgegeben, für die Fälle in welchen die Konfiguration keine eindeutige Zuordnung eines Lizenzpools zu einer Software zulässt. Das kann der Fall sein, wenn keine Zuweisung existiert oder keine eindeutige Zuordnung möglich ist.

Manuelle Administration der Lizenznutzung

Der opsi-Konfigurationseditor dokumentiert die über den opsi-Service registrierten Lizenzierungen auf der Tab-Seite "Lizenzenverwendung":

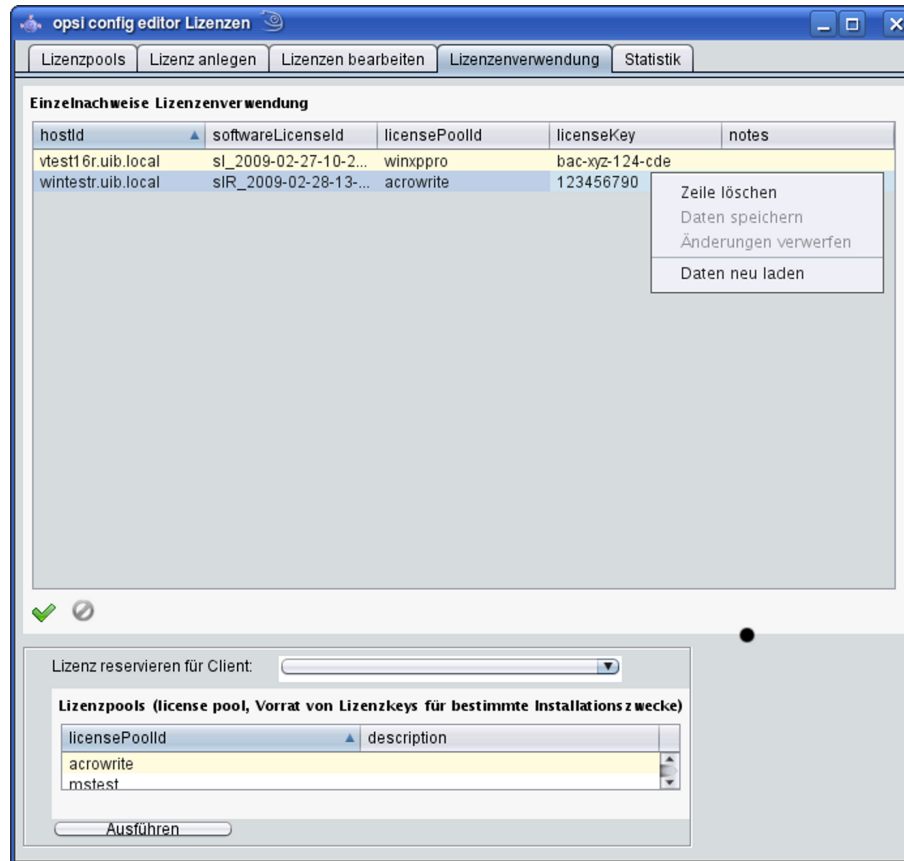


Abbildung 110: Lizenzmanagement:Tab "Lizenzenverwendung"

Die Tab-Seite ermöglicht, die Verwendung der Lizenzen auch manuell zu verwalten. Dies kann interessant sein, wenn eine Software nur vereinzelt installiert werden soll und nicht in die opsi-Verteilung eingebunden ist.

Im Einzelnen:

- Mit der Funktion "Zeilen löschen" in der Lizenzverwendungstabelle wird eine Lizenzoption wieder freigegeben.
- Der Abschnitt "Lizenz reservieren" unten auf der Seite dient dazu, eine Lizenzoption anzufordern und zu belegen.
- Durch Bearbeiten des Lizenzschlüselfeldes in der Lizenzverwendungstabelle kann der tatsächlich für eine Lizenzierung verwendete Schlüssel (neu) bestimmt werden.

Erhaltung und Löschung der Lizenzverwendungen

Wenn eine Software erneut installiert wird und der Winst mit *DemandLicenseKey* eine Lizenz anfordert, wird die vorher zugeordnete Lizenzoption weiter verwendet. Insbesondere liefert die Winst-Funktion denselben Schlüssel wie vorher.

Falls dies nicht gewünscht ist, muss die Verwendung der Lizenzierung durch den Winst mit **FreeLicense**, mit dem opsi-service-Aufruf `deleteSoftwareLicenseUsage` oder manuell aufgehoben werden.

Entsprechend bleiben bei der Reinstallation eines PCs die Lizenzverwendungen erhalten, sofern sie nicht ausdrücklich gelöscht werden. Um sie freizugeben, können auf der Tab-Seite "Lizenzenverwendung" die entsprechenden Lizenzen herausgesucht und gelöscht werden oder es kann der Serviceaufruf `deleteAllSoftwareLicenseUsages` (mit der Host-ID des betreffenden PCs als Parameter) verwendet werden.

9.9.7 Abgleich mit der Software-Inventarisierung

Die Tab-Seite "Abgleich mit der Inventarisierung" verzeichnet für jeden PC und jeden Lizenzpool, ob eine Lizenzpool-Verwendung mit dem opsi-Lizenzmanagement registriert ist (*used_by_opsi*) und ob auf dem PC laut Software-Inventarisierung (mittels *swaudit*) eine Windows-Software, die eine Lizenz aus dem Pool benötigen würde, installiert ist (*SWinventory_used*).

Damit die Ergebnisse von *swaudit* die faktischen Lizenzverwendungen beschreiben können, müssen die relevanten Windows-Software-IDs den jeweiligen Lizenzpools zugeordnet worden sein (Tab-Seite "Lizenzpools").

Das Lizenzmanagement zählt beim Abgleich mit der Softwareinventarisierung nur maximal 1 Vorkommen pro Client und Lizenzpool. Wenn also ein Lizenzpool *office2010* mit 10 verschiedenen Mustern aus der Softwareinventarisierung verknüpft ist, welche alle ein Hinweis darauf sind das hier *MS Office 2010* installiert ist, so wird das nur als eine Installation gezählt auch wenn alle 10 Muster auf einem Client gefunden werden.

hostId	licensePoolId	used_by_opsi	SWinventory_used
pctry3dettef.uib.local	p_win2k	<input type="checkbox"/>	<input type="checkbox"/>
pctry4dettef.uib.local	p_license-test-mixed	<input type="checkbox"/>	<input type="checkbox"/>
pctry4dettef.uib.local	p_license-test-retail	<input type="checkbox"/>	<input type="checkbox"/>
pctry4dettef.uib.local	p_license-test-volume	<input type="checkbox"/>	<input type="checkbox"/>
pctry4dettef.uib.local	p_winxpro	<input type="checkbox"/>	<input checked="" type="checkbox"/>
pctry4dettef.uib.local	p_winxpro-msdn	<input type="checkbox"/>	<input type="checkbox"/>
pctry5dettef.uib.local	p_license-test-retail	<input type="checkbox"/>	<input type="checkbox"/>
pctry5dettef.uib.local	p_license-test-volume	<input type="checkbox"/>	<input type="checkbox"/>
pctry5dettef.uib.local	p_win7-msdn-professional	<input type="checkbox"/>	<input type="checkbox"/>
pcuib1.uib.local	p_license-test-volume	<input type="checkbox"/>	<input type="checkbox"/>
pcuwb03.uib.local	p_win2k	<input type="checkbox"/>	<input type="checkbox"/>
pcuwb03.uib.local	p_win7-msdn-professional	<input type="checkbox"/>	<input type="checkbox"/>
pcuwb03.uib.local	p_winxpro-msdn	<input type="checkbox"/>	<input type="checkbox"/>
praktikant0.uib.local	p_winxpro-msdn	<input type="checkbox"/>	<input type="checkbox"/>
praktikant1.uib.local	p_license-test-mixed	<input type="checkbox"/>	<input type="checkbox"/>
praktikant1.uib.local	p_winxpro-msdn	<input type="checkbox"/>	<input type="checkbox"/>
rtst1-winxp.uib.local	p_winxpro-msdn	<input type="checkbox"/>	<input type="checkbox"/>
samsam.uib.local	p_winxpro-msdn	<input type="checkbox"/>	<input type="checkbox"/>
senbwf.uib.local	p_winxpro-msdn	<input type="checkbox"/>	<input type="checkbox"/>
stb-40-wks-101.uib.local	p_win2k	<input type="checkbox"/>	<input type="checkbox"/>

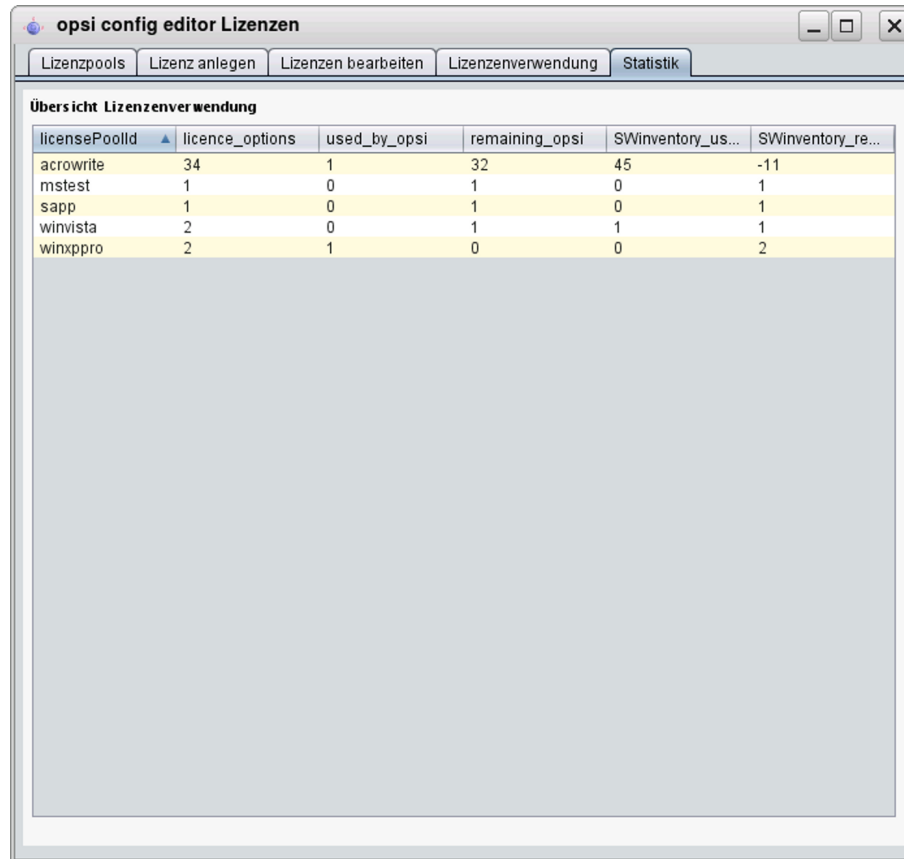
Abbildung 111: Lizenzmanagement:Tab "Abgleich mit Inventarisierung"

Die Tabelle kann wie stets per *Drag & Drop* z.B. in eine Tabellenkalkulation übernommen werden. Falls der *opsi-configed* über die entsprechenden Rechte verfügt, d.h. standalone (nicht in einer Applet-Sandbox) läuft, kann die Tabelle über eine Option des Kontextmenüs auch ausgedruckt werden.

Mittels des Configs *configed.license_inventory_extradisplayfields*, das in der Host-Parameter-Seite des Servers bearbeitet werden kann, können zusätzliche Informationen zum Client in die Tabelle aufgenommen werden.

9.9.8 Übersicht über den globalen Lizenzierungsstand

Die Tab-Seite "Statistik" dient dazu, eine summarische Übersicht über die genutzten und noch freien Lizenzoptionen der verschiedenen Lizenzpools zu erhalten.



licensePoolId	licence_options	used_by_opsi	remaining_opsi	SWinventory_us...	SWinventory_re...
acrowrite	34	1	32	45	-11
mstest	1	0	1	0	1
sapp	1	0	1	0	1
winvista	2	0	1	1	1
winxpro	2	1	0	0	2

Abbildung 112: Lizenzmanagement:Tab "Statistik"

Zusätzlich zur Angabe der registrierten Lizenzverwendungen (*used by opsi*) bzw. der hiernach noch freien (*remaining...*) Lizenzen wird in die Übersicht auch die Gesamtzahl tatsächlich vorfindbarer Installationen, die eigentlich eine Lizenz benötigen, einbezogen (*SWinventory_used*)

Die Daten der Spalte *SWinventory_used* beruhen auf den Scans der Registry der Clients, die das opsi-Produkt *swaudit* durchführt, und den Zuordnungen der hiermit ermittelten Windows-Software-IDs zu den jeweiligen Lizenzpools, verwaltet in der Tab-Seite "Lizenzpools" (vgl. Abschnitt 9.9.3).

Über eine Option des Kontextmenüs kann die Tabelle ausgedruckt werden (aufgrund der spezifischen Security-Restriktionen nicht aus dem Applet), mit *Drag & Drop* können die Daten z.B. in eine Tabellenkalkulation übernommen werden.

Fall Downgrade-Option

Wenn eine Downgrade-Option konfiguriert wurde (wie in Abschnitt 9.9.5 beschrieben), äußert sich dies in der statistischen Übersicht der Lizenzverwendung wie folgt:

Eine Downgrade-Lizenz räumt je eine Lizenzierungsoption für (mindestens) zwei Lizenzpools ein. Nur eine der beiden kann tatsächlich genutzt werden. Sobald daher eine Lizenzoption gezogen ist, verringert sich in der Spalte *remaining_opsi* in *beiden* Zeilen der Wert um je 1. Scheinbar vermindert sich also die Zahl der verfügbaren Lizenzen um 2! Dies spiegelt aber die tatsächliche Berechtigungssituation wider.

9.9.9 Service-Methoden zum Lizenzmanagement

Die Service-Methoden zum Lizenzmanagement können über das Kommandozeilenwerkzeug `opsi-admin` verwendet werden, um in einem Skript z.B. vorhandene Lizenzen aus einer Datei einzulesen.

Entsprechende Beispiele finden Sie in den Produkten *license-test-...opsi* unter <http://download.uib.de/opsi4.0/products/license-management/>. Wenn Sie diese Pakete mit `opsi-package-manager -i *.opsi` installieren, so finden Sie unter `/var/lib/opsi/depot/<produktname>` die entsprechenden Scripte: `create_license-*.sh`. Hier als Beispiel das script `create_license-mixed.sh` (ziehen Sie sich im Zweifelsfall ein aktualisiertes Skript von der genannten Adresse).

```
#!/bin/bash
# This is a test and example script
# (c) uib gmbh licensed under GPL

PRODUCT_ID=license-test-mixed
# read the license key from a file
# myretailkeys.txt has one licensekey per line
MYRETAILKEYS='cat myretailkeys.txt'
# myoemkeys.txt has one pair: <licensekey> <hostid.domain.tld> per line
MYOEMKEYS='cat myoemkeys.txt'
# some output
echo "$PRODUCT_ID"

# this is the function to create the oem licenses
#####
createlic ()
{
while [ -n "$1" ]
do
    #echo $1
    AKTKEY=$1
    shift
    #echo $1
    AKTHOST=$1
    shift
    echo "createSoftwareLicense with oem key: ${PRODUCT_ID}-oem-${AKTKEY} for host ${AKTHOST}"
    MYLIC='opsi-admin -d method createSoftwareLicense "" "c_${PRODUCT_ID}" "OEM" "1" "${AKTHOST}" ""'
    opsi-admin -d method addSoftwareLicenseToLicensePool "$MYLIC" "p_${PRODUCT_ID}" "${PRODUCT_ID}-oem-${AKTKEY}"
done
}
#####

# here the script starts

# delete the existing license pool and all connected licenses
# ATTENTION: never (!) do this on a productive system
echo "deleteLicensePool p_${PRODUCT_ID}"
opsi-admin -d method deleteLicensePool "p_${PRODUCT_ID}" true

# delete the existing license contract
echo "deleteLicenseContract c_${PRODUCT_ID}"
opsi-admin -d method deleteLicenseContract "c_${PRODUCT_ID}"

# create the new license pool
# the used method has the following syntax:
# createLicensePool(*licensePoolId, *description, *productIds, *windowsSoftwareIds)
echo "createLicensePool p_${PRODUCT_ID}"
opsi-admin -d method createLicensePool "p_${PRODUCT_ID}" "opsi license test" \'[""${PRODUCT_ID}"]\' \'[""${PRODUCT_ID}"\
']\'

# create the new license contract
# the used method has the following syntax:
# createLicenseContract(*licenseContractId, *partner, *conclusionDate, *notificationDate, *expirationDate, *notes)
echo "createLicenseContract c_${PRODUCT_ID}"
opsi-admin -d method createLicenseContract "c_${PRODUCT_ID}" "uib gmbh" "" "" "" "test contract"

# create the new license and add the key(s)
# the used methods have the following syntax:
# createSoftwareLicense(*softwareLicenseId, *licenseContractId, *licenseType, *maxInstallations, *boundToHost, *\
expirationDate)
# addSoftwareLicenseToLicensePool(softwareLicenseId, licensePoolId, *licenseKey)

# create the retail licenses:
```

```

for AKTKEY in $MYRETAILKEYS
do
    echo "createSoftwareLicense with retail key: ${PRODUCT_ID}-retail-${AKTKEY}"
    MYLIC='opsi-admin -dS method createSoftwareLicense "" "c_${PRODUCT_ID}" "RETAIL" "1" "" ""'
    opsi-admin -d method addSoftwareLicenseToLicensePool "$MYLIC" "p_${PRODUCT_ID}" "${PRODUCT_ID}-retail-${AKTKEY}"
done

# create the oem licenses
createlic $MYOEMKEYS

# create the volume licenses
echo "createSoftwareLicense with volume key: ${PRODUCT_ID}-vol-key"
MYLIC='opsi-admin -dS method createSoftwareLicense "" "c_${PRODUCT_ID}" "VOLUME" "10" "" ""'
opsi-admin -d method addSoftwareLicenseToLicensePool "$MYLIC" "p_${PRODUCT_ID}" "${PRODUCT_ID}-vol-key" #

```

9.9.10 Beispielprodukte und Templates

Im Downloadbereich von uib finden sich unter

<http://download.uib.de/opsi4.0/products/license-management/>

vier Beispielprodukte: je ein Produkt zur Verwendung von Retail, OEM und Volumenlizenzen sowie ein Produkt, welches alle drei Lizenztypen vereint.

Diese Produkte belegen bei der Installation beispielhaft Lizenzen und geben sie bei der Deinstallation wieder frei. Weiterhin hinterlassen diese Beispielprodukte auch entsprechende Spuren in der Softwareinventarisierung, die vom Lizenzmanagement zum Abgleich verwendet werden können. Alle diese Produkte enthalten (wie oben schon erwähnt) ein Shell-Skript, mit dem zu Testzwecken automatisiert die Lizenzpools, Lizenzverträge und Lizenzen angelegt werden können.

Das Standardtemplate für Winst-Skripte *opsi-template* enthält ebenfalls die notwendigen Beispiele zur Nutzung des opsi-Lizenzmanagements.

9.10 opsi WAN/VPN-Erweiterung

Die WAN/VPN-Erweiterung bietet opsi-Administratoren die Möglichkeit, auch Clients hinter langsamen Leitungen in opsi einzubinden. Diese Dokumentation soll die Funktionsweise dieser Erweiterung von opsi erläutern und einen Leitfaden bieten, wie man diese Erweiterung konfigurieren und pflegen kann. Beachten Sie bitte, dass zur Zeit die gleichzeitige Nutzung von WAN-Erweiterung und Installation bei Shutdown auf einem Client noch nicht freigegeben ist. Auf einem opsi-Server auf verschiedenen Clients können Sie beide Erweiterungen natürlich nutzen.

9.10.1 Vorbedingungen für die WAN/VPN-Erweiterung

Als Erstes sei an dieser Stelle erwähnt, dass dieses Modul momentan eine [kofinanzierte opsi Erweiterung](#) ist. Weitere Details hierzu finden Sie in Abschnitt [9.1](#).

Es sind eine Reihe von Vorbedingungen nötig, um dieses Modul einsetzen zu können. Zunächst werden *Produktgruppen* benötigt, diese stehen erst ab opsi 4.0 zur Verfügung. Weiterhin werden die Pakete *opsi-client-agent* und *opsi-configed* ab Version 4.0.1 benötigt.

Tabelle 19: Benötigte Pakete

opsi-Paket	Version
opsi-client-agent	>=4.0.1-1
opsi-winst	>=4.10.8.12
python-opsi	>=4.0.1-7
opsi-configed	>=4.0.1.6-1

9.10.2 Überblick über die WAN/VPN-Erweiterung

Grob betrachtet verläuft die Softwareverteilung per opsi in der Regel folgendermaßen ab:

- Der *opsi-Loginblocker* blockiert beim Systemstart die Anmeldung von Benutzern am System.
- Der *opsiclientd* nimmt Kontakt zum *opsi-configserver* auf.
- Sind *Produktaktionen* für den Client gesetzt, verbindet dieser ein Netzlaufwerk mit dem *opsi-depot*.
- Der *opsi-winst* wird gestartet und nimmt ebenfalls Kontakt zum *opsi-configserver* auf.
- Der *opsi-winst* bearbeitet die gesetzten *Produktaktionen*, wobei er direkt auf das Netzlaufwerk zugreift.
- Benötigte Reboots werden ausgeführt und der Prozess beginnt erneut.
- Nach dem Abschluss aller *Produktaktionen* werden Log-Dateien an den *opsi-configserver* übertragen und die Anmeldung für den Anwender freigegeben.

Betrachten wir nun den Fall eines Clients in einer Außenstelle, die über eine WAN-Leitung an das LAN angebunden ist, in dem sich *opsi-configserver* und *opsi-depotserver* befinden:

- Bei der Kommunikation mit dem *opsi-configserver* werden nur geringe Datenmengen übertragen, hier tritt keine problematische Verzögerung des Softwareverteilungs-Prozesses auf.
- Das Bearbeiten der *Produktaktionen*, dauert jedoch je nach Paket-Größe, Bandbreite und Latenz der WAN-Verbindung, sehr lange. Auch kann es bei Dateizugriffen zu Timeouts kommen.
- Der Rechner ist dementsprechend lange für den Anwender blockiert.

Sollte sich das Aufstellen eines eigenen *opsi-depotserver* in der Außenstelle nicht rentieren, kann das Problem über die Verwendung der WAN/VPN-Erweiterung gelöst werden.

Der *opsi-client-agent* kann hierbei folgendermaßen konfiguriert werden:

- Beim Systemstart findet bei einem ungefülltem Produkt-Cache keine Softwareverteilung statt. Der Login wird nicht weiter blockiert.
- Bei gesetzten *Produktaktionen* beginnt der *opsiclientd* mit der Übertragung der benötigten Dateien vom *opsi-depot* auf den lokalen Rechner. Die Übertragung kann hierbei in der Bandbreite beschränkt und auch je nach aktueller Netz-Auslastung dynamisch angepasst werden.
- Nach abgeschlossener Synchronisation der Produkt-Pakete mit dem lokalen Cache wird eine Reboot-Anforderung ausgelöst.
- Der angemeldete Benutzer stimmt dem geforderten Reboot zu oder der Rechner wird später aus einem anderen Grund neu gestartet.
- Beim Systemstart wird ein gefüllter Produkt-Cache festgestellt und die Softwareverteilung findet wie gewohnt statt. Hierbei wird jedoch mit den lokalen Dateien gearbeitet. Die Installation läuft somit sogar schneller als im LAN.

Betrachten wir nun den Fall eines Notebooks, das in vielen Fällen beim Systemstart überhaupt keinen Kontakt zum *opsi-configserver* herstellen kann:

- Ein Kontakt-Aufbau zum *opsi-configserver* beim Systemstart läuft in den meisten Fällen in einen Timeout.
- Unter Umständen kann der Kontakt zum *opsi-configserver* erst dann hergestellt werden, wenn sich ein Benutzer am System anmeldet und über einen VPN-Adapter eine Verbindung zum Unternehmens-Netzwerk herstellt.

- Ohne Verbindung zum *opsi-configserver* kann keine Softwareverteilung stattfinden.

Auch dieses Problem kann über die Verwendung der WAN/VPN-Erweiterung gelöst werden. Der *opsi-client-agent* kann hierbei folgendermaßen konfiguriert werden:

- Beim Systemstart findet bei einem ungefülltem Produkt- oder Config-Cache keine Softwareverteilung statt. Der Login wird nicht weiter blockiert.
- Bei Aktivierung eines Netzwerk-Adapters und/oder in regelmäßigen Zeitabständen wird im Hintergrund versucht, eine Verbindung zum *opsi-configserver* herzustellen.
- Ist der *opsi-configserver* erreichbar, beginnt der *opsiclientd* mit:
 - Der Synchronisation der Konfigurationen.
 - Der Übertragung der benötigten Dateien vom *opsi-depot* auf den lokalen Rechner. In Verbindung mit der opsi-Erweiterung *Dynamische Depot-Auswahl* findet die Datei-Übertragung immer von dem *opsi-depot* statt, zu dem die beste Netzwerkverbindung besteht.
- Nach abgeschlossener Synchronisation der Produkt-Pakete und der Konfigurationen mit dem lokalen Cache wird eine Reboot-Anforderung ausgelöst.
- Der angemeldete Benutzer stimmt dem geforderten Reboot zu oder der Rechner wird später aus einem anderen Grund neu gestartet.
- Beim Systemstart wird ein gefüllter Produkt- und Config-Cache festgestellt. Die Softwareverteilung findet wie gewohnt statt. Hierbei wird jedoch mit den lokalen Dateien und den lokalen Konfigurationen gearbeitet. Der *opsiclientd* übernimmt hierfür die Funktionen des *opsi-configservers* und des *opsi-depotserver*s.
- Beim nächsten Verbindungsaufbau zum *opsi-configserver* werden die Ergebnisse (die Änderungen an den Konfigurationen, Log-Dateien ...) synchronisiert.

Der Mechanismus zur Produkt-Synchronisation kann hierbei mehrfach unterbrochen werden. Die Datei-Synchronisation setzt immer wieder am Punkt der Unterbrechung an. Bereits übertragene Daten müssen nicht erneut übertragen werden.

Da die WAN/VPN-Erweiterung die Möglichkeit eröffnet, auch Clients an opsi anzubinden, die sich außerhalb eines geschützten Firmen-Netzwerks befinden, sind zusätzliche Sicherheitsmaßnahmen bei der Kommunikation zwischen Clients und Servern zu empfehlen.

So bietet der *opsiclientd* nun die Möglichkeit, die Identität eines *opsi-server*s zu verifizieren. Hierfür wird das Keypair des SSL-Zertifikats des *opsiconfd* verwendet.

Über diesen Mechanismus können sowohl *opsi-configserver* als auch *opsi-depotserver* verifiziert werden, jedoch nur wenn die Kommunikation über den *opsiconfd* und per *SSL* erfolgt. Im Falle eines *opsi-depot*s muss der Datei-Zugriff also über den *opsiconfd* per *HTTPS/WEBDAVS* erfolgen. Der Zugriff per *CIFS/SMB* wird nicht überprüft.

9.10.3 Caching von Produkten

Das Cachen von *Produkten* übernimmt der *ProductCacheService*, der Bestandteil des *opsiclientd* ist.

Der *ProductCacheService* synchronisiert die lokalen Kopien der in einem *Produkt* enthaltenen Dateien mit den Dateien des *Produkts* auf einem *opsi-depot*. Das Basis-Verzeichnis des Produkt-Caches ist konfigurierbar und standardmäßig auf `%SystemDrive%\opsi.org\cache\depot` gesetzt.

Protokoll zum Zugriff auf ein opsi-depot

Bei der Übertragung von Produkt-Dateien werden zwei Protokolle unterstützt.

- *CIFS/SMB*
- *HTTP(S)/WEBDAV(S)*

Bei der Verwendung von *CIFS/SMB* wird eine Verbindung zu der *depotRemoteUrl* hergestellt, die in den Eigenschaften eines *opsi-depots* konfiguriert ist. Im Falle von *HTTP(S)/WEBDAV(S)* wird die ebenfalls am Depot konfigurierte *depotWebdavUrl* verwendet.

Welches Protokoll verwendet wird, kann über das *Hostparameter* `clientconfig.depot.protocol` Client-spezifisch konfiguriert werden. Die möglichen Werte sind `cifs` und `webdav`.

Anmerkung

Auch das opsi-linux-bootimage wertet diese Konfiguration aus und verwendet das angegebenen Protokoll.

Die .files-Datei

Basis für die Synchronisation ist die Datei `<product-id>.files`, die im Basis-Verzeichnis eines *Produkts* auf dem *opsi-depot* zu finden ist. Die Datei enthält Informationen zu allen in einem *Produkt* enthaltenen Dateien, Verzeichnissen und symbolischen Links. Jede Zeile in der Datei entspricht einer solchen Information. Die einzelnen Informations-Typen werden durch ein Leerzeichen voneinander getrennt.

Das erste Zeichen in einer Zeile gibt den Typ des Eintrags an, mögliche Werte sind:

- `d` für ein Verzeichnis
- `f` für eine Datei
- `l` für einen symbolischen Link

Abgetrennt durch ein Leerzeichen folgt der relative Pfad in einfachen Anführungszeichen.

Der nächste Eintrag entspricht der Dateigröße (bei Verzeichnissen und Links steht hier eine 0).

Im Falle einer Datei folgt noch die MD5-Summe der Datei, bei einem symbolischen Link das Ziel des Links.

Auszug einer *.files*-Datei:

```
d 'utils' 0
f 'utils/patch_config_file.py' 2506 d3007628addf6d9f688eb4c2e219dc18
l 'utils/link_to_patch_config_file.py' 0 '/utils/patch_config_file.py'
```

Die *.files*-Datei wird beim Einspielen von *Produkt-Paketen* (nach dem Lauf des *postinst*-Skriptes) automatisch erzeugt.



Warnung

Bei Verwendung der WAN/VPN-Erweiterung sollten die Dateien auf einem *opsi-depot* nicht manuell bearbeitet werden, da sonst die in der *.files*-Datei enthalten Informationen nicht mehr zutreffen und dies zu Fehlern bei der Synchronisation führt.

Ablauf des Produkt-Cachings

Die Synchronisation einer lokalen Kopie eines *Produkts* läuft folgendermaßen ab:

- Die *.files*-Datei des *Produkts* wird auf den lokalen Rechner übertragen.
- Es wird geprüft ob genügend freier Speicherplatz für das Caching vorhanden ist. Sollte der verfügbare Speicherplatz nicht ausreichen wird durch das Löschen von *Produkten* Platz geschaffen. Hierbei werden bevorzugt *Produkte* gelöscht, die seit längerem nicht mehr benötigten (synchronisiert) wurden.
- Das Cache-Verzeichnis, das die lokale Kopie enthält, wird angelegt sofern es noch nicht existiert.
- Anhand der Einträge in der *.files*-Datei werden nicht mehr benötigte Dateien und Verzeichnisse aus dem Cache-Verzeichnis entfernt.

- Die *.files*-Datei wird nun der Reihe nach durchgearbeitet.
 - Ein fehlendes Verzeichnis wird angelegt.
 - Eine fehlende Datei wird übertragen.
 - Vorhandene Dateien werden anhand der Größe und MD5-Summe überprüft und bei Abweichungen (teilweise) neu übertragen.

Das Ergebnis der Synchronisation ist eine exakte Kopie des Produkt-Verzeichnisses auf dem *opsi-depot*.

Anmerkung

Unter Windows werden keine Symbolischen Links erzeugt, statt eines Links wird eine Kopie des Link-Ziels angelegt.

Ein erfolgreich abgeschlossenes Produkt-Caching hat zur Folge, dass:

- Der Zustand `products_cached` den Wert `true` annimmt und dieser auch über einen Neustart hinweg erhalten bleibt (siehe: Abschnitt 6.1.3).
- Ein Event vom Typ `sync completed` ausgelöst wird.

Konfiguration des Produkt-Cachings

Die allgemeine Konfiguration des Produkt-Cachings wird in der `opsiclientd.conf` innerhalb der Sektion `[cache_service]` vorgenommen.

- `product_cache_max_size` (integer): Die maximale Größe des Produkt-Caches in Bytes. Hiermit wird sichergestellt, dass der durch das Produkt-Caching belegte Speicherplatz die konfigurierte Größe nicht überschreitet.
- `storage_dir` (string): Der Pfad zum Verzeichnis, in dem das Basis-Verzeichnis `depot` für das Produkt-Caching angelegt wird.

Weitere Konfigurationen erfolgen Event-spezifisch.

Innerhalb einer Event-Konfigurations-Sektion `[event_<event-config-id>]` existieren folgende Optionen:

- `cache_products` (boolean): Steht der Wert dieser Option auf `true` beginnt der *ProductCacheService* beim Auftreten des Events mit dem Cachen von *Produkten*, für die eine *Produktaktion* gesetzt ist. Ist zusätzlich der Wert der Option `use_cached_products` auf `true` gesetzt, wird die weitere Bearbeitung des Events solange verzögert, bis das Cachen der *Produkte* abgeschlossen ist.
- `cache_max_bandwidth` (integer): Die maximale Bandbreite in Byte/s, die beim Cachen verwendet werden soll. Bei einem Wert kleiner oder gleich 0 wird keine Bandbreiten-Begrenzung vorgenommen.
- `cache_dynamic_bandwidth` (boolean): Steht der Wert dieser Option auf `true`, wird die für die Übertragung verwendete Bandbreite dynamisch angepasst. Hierbei wird der Netzwerk-Verkehr auf der Netzwerkschnittstelle zum *opsi-depot* kontinuierlich überwacht. Wird dabei Netzwerk-Verkehr festgestellt, der nicht durch das Produkt-Caching entsteht, wird die Bandbreite der Übertragung stark reduziert, um andere Anwendungen möglichst wenig zu beeinflussen. Ist die Bandbreite dynamisch reduziert und der Netzwerk-Verkehr im Wesentlichen auf das Produkt-Caching zurückzuführen, wird die dynamische Begrenzung wieder aufgehoben. Der Wert von `cache_max_bandwidth` wird auch bei Verwendung der dynamischen Bandbreiten-Begrenzung weiterhin berücksichtigt.
- `use_cached_products` (boolean): Ist dieser Wert auf `true` gesetzt, wird beim Bearbeiten der *Produktaktionen* der lokale Produkt-Cache verwendet. Ist das Caching der *Produkte* zu diesem Zeitpunkt noch nicht abgeschlossen, wird die Bearbeitung des Events mit einem Fehler beendet.

9.10.4 Caching von Konfigurationen

Das Cachen von Konfigurationen übernimmt der *ConfigCacheService*, der Bestandteil des *opsi-clientd* ist.

Der *ConfigCacheService* synchronisiert ein lokales *Client-Cache-Backend* mit dem *Config-Backend* des *opsi-configservers*.

Der *opsi-clientd* bietet per *WebService* einen Zugriff auf das Backend und stellt somit eine ähnliche Funktionalität wie der *opsiconfd* bereit.

Das lokale *Client-Cache-Backend*

Das lokale *Client-Cache-Backend* basiert auf *SQLite* und besteht im Wesentlichen aus einer Arbeitskopie, einem Snapshot und einem Modification-Tracker, der über Änderungen an der Arbeitskopie Buch führt.

Das Basis-Verzeichnis des Config-Caches ist konfigurierbar und standardmäßig auf `%SystemDrive%\opsi.org\cache\config` gesetzt. Der Snapshot entspricht dem Stand der Konfigurationen auf dem *opsi-configserver* zum Zeitpunkt der letzten Synchronisation.

Die Arbeitskopie entspricht zu Beginn dem Snapshot und wird im Laufe der Aktionen modifiziert.

Ablauf der Synchronisation von Konfigurationen

Die Synchronisation der lokalen Änderungen im *Client-Cache-Backend* mit dem *Config-Backend* des *opsi-configservers* läuft folgendermaßen ab:

- Die im Modification-Tracker registrierten Änderungen an der Arbeitskopie werden auf den *opsi-configserver* übertragen. Änderungen an den Konfigurationen auf dem *opsi-configserver* seit der letzten Synchronisation werden durch Vergleich mit dem Snapshot erkannt. Kommt es bei der Rückübertragung der Modifikationen zu Konflikten greifen folgende Regeln:
 - Im Fall von Inventarisierungsdaten besitzen die Daten des Clients Priorität
 - Bei *Action-Requests* gilt der Wert des *opsi-configservers*
 - Im Fall von *Installationsstatus* und *Aktionsergebnis* wird der Client-Wert bevorzugt.
 - Ist die Verwendung des opsi-Lizenzmanagement eingeschaltet (config: *license-management.use=true*), so wird versucht über die Kopplung *opsi-Produkt* zu *opsi-Lizenzpool* eine freie Lizenz zu reservieren. Diese verwendeten Software-Lizenzen wird mit repliziert. Bei der Replikation reservierte, ungenutzte Lizenzen werden wieder freigegeben.
 - Der *opsi-configserver* behält beim Zustand von *Hostparametern* und *Product-Properties* recht.
- Der Modification-Tracker wird geleert.
- Die Log-Dateien werden übertragen.

Die Replikation des *Config-Backend* des *opsi-configservers* in das *Client-Cache-Backend* läuft folgendermaßen ab:

- Die Replikation findet nur statt, wenn auf dem *opsi-configserver* *Action-Requests* gesetzt sind, die *Produktaktion* **always** gilt hierbei als nicht gesetzt. Ist der Zustand der *Action-Requests* seit dem letzten Replikations-Lauf unverändert, findet ebenfalls keine Replikation statt.
- Der Modification-Tracker die Arbeitskopie und der Snapshot werden geleert.
- Die zum autarken Arbeiten benötigten Konfigurationen werden repliziert.
- Sind *Action-Requests* für *Produkte* gesetzt die als lizenzpflichtig markiert wurden, wird eine Software-Lizenz aus einem, dem *Produkt* zugeordneten, *Lizenzpool* reserviert.
- Zusätzlich benötigte Daten, wie `auditHardwareConfig` und `modules` werden übertragen.
- Der Snapshot und die Arbeitskopie werden auf den gleichen Stand gebracht.

Eine erfolgreiche Replikation vom Server zum Client hat zur Folge, dass:

- Der Zustand `config_cached` den Wert `true` annimmt und dieser auch über einen Neustart hinweg erhalten bleibt (siehe: Abschnitt 6.1.3).
- Ein Event vom Typ `sync completed` ausgelöst wird.

Konfiguration des Config-Cachings

Die Konfiguration des Config-Cachings erfolgt hauptsächlich Event-spezifisch.

Innerhalb einer Event-Konfigurations-Sektion [`event_<event-config-id>`] existieren folgende Optionen:

- `sync_config_to_server` (boolean): Steht der Wert dieser Option auf `true`, beginnt der *ConfigCacheService* beim Auftreten des Events die im Modification-Tracker registrierten Änderungen zum *opsi-configserver* zu übertragen. Das Ergebnis dieser Aktion wird in jedem Fall abgewartet.
- `sync_config_from_server` (boolean): Ist dieser Wert auf `true` gesetzt, beginnt der *ConfigCacheService* mit der Replikation. Ist zusätzlich der Wert der Option `use_cached_config` auf `true` gesetzt wird die weitere Bearbeitung des Events solange verzögert, bis die Replikation abgeschlossen ist.
- `use_cached_config` (boolean): Steht der Wert dieser Option auf `true`, wird beim Bearbeiten der *Produktaktionen* das *Client-Cache-Backend* verwendet. Ist die Synchronisation zu diesem Zeitpunkt noch nicht abgeschlossen wird die Bearbeitung des Events mit einem Fehler beendet.
- `post_sync_config_to_server` (boolean): Entspricht `sync_config_to_server`, wird jedoch nach Abschluss der *Produktaktionen* ausgewertet.
- `post_sync_config_from_server` (boolean): Entspricht `sync_config_from_server`, wird jedoch nach Abschluss der *Produktaktionen* ausgewertet.

9.10.5 Empfohlene Konfiguration bei Verwendung der WAN/VPN-Erweiterung

Das *opsi-client-agent*-Paket bringt eine, für die WAN/VPN-Erweiterung, vorbereitete `opsiclientd.conf` mit.

Um die WAN/VPN-Erweiterung zu aktivieren ist es lediglich notwendig, einige Events zu aktivieren und andere zu deaktivieren.

Da die Konfiguration des *opsi-client-agents* auch zentral über den Webservice erfolgen kann (siehe: Abschnitt 6.1.3), ist zu empfehlen die folgenden *Hostparameter* anzulegen.

- `opsiclientd.event_gui_startup.active` (boolean, default: `true`)
- `opsiclientd.event_gui_startup{user_logged_in}.active` (boolean, default: `true`)
- `opsiclientd.event_net_connection.active` (boolean, default: `false`)
- `opsiclientd.event_timer.active` (boolean, default: `false`)

Über diese *Hostparameter* können dann Events Client-spezifisch aktiviert bzw. deaktiviert werden. Die *Hostparameter* können über den *opsi-configd* oder *opsi-admin* angelegt werden.

Zum Anlegen der *Hostparameter* über *opsi-admin* sind die folgenden Befehle auf dem *opsi-configserver* auszuführen:

```
opsi-admin -d method config_createBool opsiclientd.event_gui_startup.active "gui_startup active" true
opsi-admin -d method config_createBool opsiclientd.event_gui_startup{user_logged_in}.active "gui_startup{user_logged_in}\
} active" true
opsi-admin -d method config_createBool opsiclientd.event_net_connection.active "event_net_connection active" false
opsi-admin -d method config_createBool opsiclientd.event_timer.active "event_timer active" false
```

Die gesetzten Standard-Werte entsprechen hierbei den Standard-Werten der mitgelieferten `opsiclientd.conf`.

**Achtung**

Wenn Sie vorgenannten Operation zum setzen der defaults **nicht** ausführen und nur die nachfolgenden, dann stellen Sie **alle** Clients auf WAN um !

Für einen WAN/VPN-Client, der Konfigurationen und Produkte cachen soll, werden die *Hostparameter* wie folgt konfiguriert:

- `opsiclientd.event_gui_startup.active: false`
- `opsiclientd.event_gui_startup{user_logged_in}.active: false`
- `opsiclientd.event_net_connection.active: true`
- `opsiclientd.event_timer.active: true`

Die Client-spezifischen *Hostparameter* können über den *opsi-configed* oder *opsi-admin* gesetzt werden.

Zum Setzen der *Hostparameter* über *opsi-admin* sind die folgenden Befehle auf dem *opsi-configserver* auszuführen (im Beispiel für einen Client mit der opsi-host-Id `vpnclient.domain.de`):

```
opsi-admin -d method configState_create opsiclientd.event_gui_startup.active vpnclient.domain.de false
opsi-admin -d method configState_create opsiclientd.event_gui_startup{user_logged_in}.active vpnclient.domain.de false
opsi-admin -d method configState_create opsiclientd.event_net_connection.active vpnclient.domain.de true
opsi-admin -d method configState_create opsiclientd.event_timer.active vpnclient.domain.de true
```

Diese Konfiguration hat zur Folge, dass:

- Beim Start des Rechners kein Verbindungsaufbau zum *opsi-configserver* stattfindet.
- Beim Aktivieren einer beliebigen Netzwerk-Schnittstelle ein Verbindungsaufbau zum *opsi-configserver* versucht und mit der Synchronisation im Hintergrund begonnen wird.
- Ein `timer`-Event aktiviert wird, dass in regelmäßigen Abständen aktiv wird und ebenso einen Synchronisations-Versuch unternimmt.

Wahl des Protokolls für das Caching der Produkte

Das Caching der *Produkte* kann über die Protokolle *HTTPS/WEBDAVS* oder *CIFS/SMB* erfolgen.

Bei Verwendung von *webdav* erfolgt der Zugriff auf das *opsi-depot* über den *opsiconfd*.

- Vorteile:
 - Einfache Firewall-Konfiguration, lediglich Zugriff auf Port 4447 notwendig.
 - Prüfung des SSL-Zertifikats des *opsi-depots* möglich.
- Nachteile:
 - Der *opsiconfd* erzeugt höhere Lasten auf dem opsi-depot.

Bei Verwendung von *cifs* erfolgt der Zugriff auf das *opsi-depot* über *SAMBA*.

- Vorteile:
 - Der *SAMBA*-Server ist performant, ressourcenschonend und gut skalierbar.
- Nachteile:

- Aufwändigere Firewall-Konfiguration, Zugriff auf SAMBA-Ports notwendig.
- Prüfung des SSL-Zertifikats des *opsi-depots* nicht möglich.

Eine Anleitung zur Konfiguration des Protokolls finden sich im Kapitel Abschnitt 9.10.3.

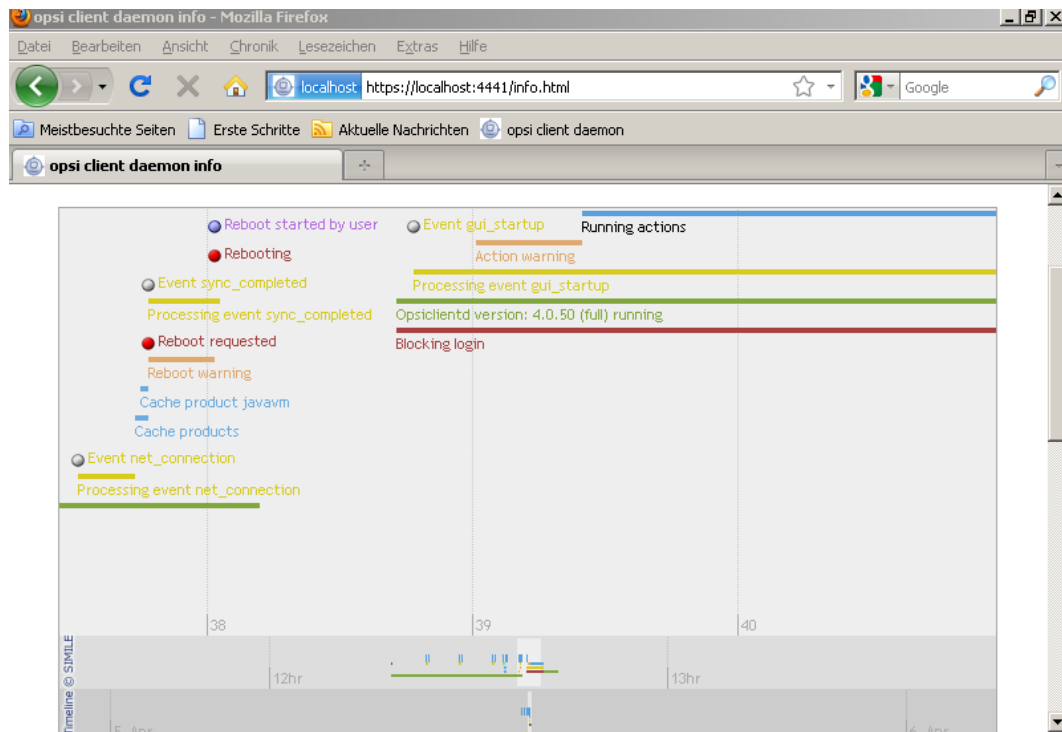


Abbildung 113: Ablauf einer Installation mit der WAN-Erweiterung in der opsiclientd-infopage

Prüfung der Server-Zertifikate

Um die Prüfung von SSL-Zertifikaten zu aktivieren, ist in der *opsiclientd.conf* innerhalb der Sektion `[global]` die Option `verify_server_cert` auf `true` zu setzen. Dies hat zur Folge, dass bei einem Verbindungsaufbau zu einem *opsiconfd* der opsi-server anhand des SSL-Zertifikats überprüft wird. Die Server-Zertifikate werden auf dem Client im Verzeichnis `c:\opsi.org\opsiclientd\server-certs` abgelegt. Der Dateiname des Zertifikats setzt sich aus der Server-Adresse (IP oder Name) und der Dateiendung `.pem` zusammen. Sollte beim Verbindungsaufbau kein gespeichertes Zertifikat gefunden werden, findet keine Überprüfung statt.

Tipp

Um ein geändertes Zertifikat neu zu publizieren, muss das auf den Clients vorhandene Zertifikat gelöscht werden. Hierfür steht auch die RPC-Methode `deleteServerCerts` bereit, die über das Control-Interface des *opsiclientd* aufgerufen werden kann.

9.11 opsi-Nagios-Connector

9.11.1 Einführung

Neben dem Client Management ist das Monitoring der IT-Infrastruktur eine der Kernfunktion in der heutigen IT. opsi ist ein Client-Management Werkzeug. Für das Monitoring gibt es erprobte und bekannte Opensource Lösungen. Daher der Ansatz opsi nicht um ein Monitoring zu erweitern, sondern eine Schnittstelle zu bestehenden Lösungen anzubieten. Mit dieser Erweiterung von opsi wird eine Schnittstelle für die bekannte Monitoring Lösung Nagios zur

Verfügung gestellt. In den folgenden Kapiteln wird der opsi-Nagios-Connector erläutert und Beispielkonfigurationen vorgestellt.

Der opsi-Nagios-Connector ist nicht fest an eine Monitoringsoftware gebunden. Programmiert wurde die Schnittstelle für Nagios/Icinga. Der Einsatz mit anderen Monitoringlösungen ist denkbar, wird momentan aber weder getestet noch unterstützt.

Die folgende Dokumentation beschreibt wie der opsi-Nagios-Connector aufgebaut ist und wie man ihn konfigurieren muss. Es wird vorausgesetzt, dass eine funktionierende Installation von Nagios bzw. Icinga vorliegt.

9.11.2 Vorbedingungen

Vorbedingungen bei opsi-Server und -Client

Dieses Modul ist momentan eine [kofinanzierte opsi Erweiterung](#).

Es sind eine Reihe von Vorbedingungen nötig, um dieses Modul einsetzen zu können. Das bedeutet, dass Sie zum Einsatz eine Freischaltdatei benötigen. Diese Freischaltung erhalten Sie beim Kauf der Erweiterung. Zu Evaluierungszwecken stellen wir Ihnen auch eine zeitlich befristete Freischaltung kostenlos zur Verfügung (→ mail an info@uib.de).

Weitere Details hierzu finden Sie in Abschnitt [9.1](#).

Technische Voraussetzungen sind opsi 4.0.2 mit den Paketständen:

Tabelle 20: Benötigte Pakete

opsi-Paket	Version
opsi-client-agent	>=4.0.2-1
opsiconfd	>=4.0.2.1-1
python-opsi	>=4.0.2.1-1

Vorbedingungen beim Nagios Server

Vorausgesetzt wird eine Nagios Installation in einer aktuellen 3.x Version oder eine Icinga Installation mindestens in der Version 1.6. Der opsi-Nagios-Connector sollte auch mit allen Nagios-Kompatiblen Monitoring-Lösungen benutzbar sein, getestet wird diese opsi Erweiterung allerdings nur gegen Nagios und Icinga. Sollte eine grafische Darstellung der opsiconfd-Performance Werte gewünscht werden, wird zusätzlich noch eine Installation des Addons pnp4nagios benötigt.

Für weitere Informationen siehe auch:

- nagios.org
- icinga.org
- pnp4nagios.org

9.11.3 Konzept

Der opsi-Nagios-Connector besteht hauptsächlich aus zwei verschiedenen Komponenten. Im Folgenden werden beide Komponenten erläutert.

opsi-Webservice Erweiterung

Eines der Hauptschwerpunkte des opsi-Nagios-Connectors ist eine Erweiterung des opsi-Webservice. Die Webservice-schnittstelle bietet damit die Möglichkeit Checks vom opsi-Server anzufordern. Das bedeutet der Nagios Server verwendet die erweiterten Webservicesmethoden zum Auslösen und Abfragen von Checks. Der Vorteil dieser Lösung ist,

dass die eigentliche Checkausführung im opsi-System ausgeführt wird und somit der Aufwand auf den überwachten Systemen gering bleibt.

Diese Erweiterung stellt eine Sammlung von Checks zur Verfügung, die in einem späteren Kapitel einzeln vorgestellt werden. Hauptsächlich wurden opsi spezifische Checks eingebaut. "Normale" Systemchecks werden nicht über den opsi-Nagios-Connector angeboten und müssen auf konventionelle Weise ausgeführt werden.

opsi-client-agent Erweiterung

Ein weiterer Teil des opsi-Nagios-Connectors ist eine Erweiterung des opsi-client-agent. Da in einer opsi-Umgebung jeder Managed-Client einen laufenden opsi-client-agent hat, bietet es sich an, diesen als Nagios-Agent zu benutzen. Wichtig bei dieser Erweiterung ist zu wissen, dass nicht alle Funktionen eines Nagios-Agenten, wie z.B. `NSClient++` implementiert wurden.

Die Möglichkeiten des opsi-client-agent umfassen das Ausführen von Pluginbefehlen auf der Kommandozeile und das Zurückliefern der Ergebnisse.

Für Situationen in denen erweiterte Funktionen vom Nagios-Agent wie NSCA benötigt werden, wird ein opsi-Paket für die Verteilung und Pflege des `NSClient++` per opsi bereitgestellt.

Der Vorteil der Verwendung des opsi-client-agent ist zum Einen, dass kein zusätzlicher Agent benötigt wird und das der Monitoring-Server keine Zugangsdaten vom Client haben muss, da die Checks zum Client alle über den `opsiconfd` laufen. Dies vereinfacht auch die Konfiguration auf der Monitoring Seite um einiges.

9.11.4 opsi-Checks

Im folgenden Kapitel werden Zweck und Konfiguration der einzelnen opsi-Checks erklärt.

Hintergrund zum richtigen Verteilen der Checks

In der allgemeinen Monitoring Sprache wird zwischen aktiven und passiven Checks unterschieden. Die Bedeutung besagt, ob Nagios/Icinga die Checks selber auslöst und ausführt (aktiv) oder ob die Hosts die Checks eigenständig ausführen und die Ergebnisse an Nagios/Icinga selbstständig zurückmelden (passiv).

Auch bei der opsi-Nagios-Connector Erweiterung gibt es diese zwei Unterscheidungen. Allerdings werden diese bei opsi als direkte- bzw. indirekte-Checks bezeichnet:

- Direkt: Diese Checks werden auf dem Client ausgeführt und liefern Ergebnisse vom Client an den Monitoring Server.
- Indirekt: Die Erhebung der Daten, die für diese Art von Checks relevant sind, findet auf Basis von Backend-Daten im opsi-System statt. Diese können von realen Situationen und Zuständen abweichen.

Ein gutes Beispiel für einen indirekten Check ist `check-opsi-client-status`. Dieser Check bezieht sich eigentlich auf einen Client, ausgeführt wird er aber auf dem opsi-Backend. Der Client-Status ist in diesem Fall der Zustand des Clients aus Sicht von opsi. Im Gegensatz dazu wird jeder Check der tatsächlich am Client ausgeführt wird als direkter Check bezeichnet.

Die richtige Verteilung der Checks und damit die richtige Konfiguration erfordert zunächst die Analyse der eigenen opsi-Umgebung. Da opsi sehr flexibel einsetzbar ist, können verschiedene Szenarien auftreten. Hier werden die am häufigsten mit opsi eingesetzten Installationsvarianten abgehandelt. (Für spezielle Installation sollten Sie uns kontaktieren oder direkt über einen bestehenden Supportvertrag Ihre Situation schildern.)

Ein opsi-Server (Standalone Installation): Diese Variante ist die am häufigsten eingesetzten Variante. Bei dieser Installation ist der Configserver auch gleichzeitig der Depotserver und somit werden alle Checks über den opsi-Configserver aufgerufen.

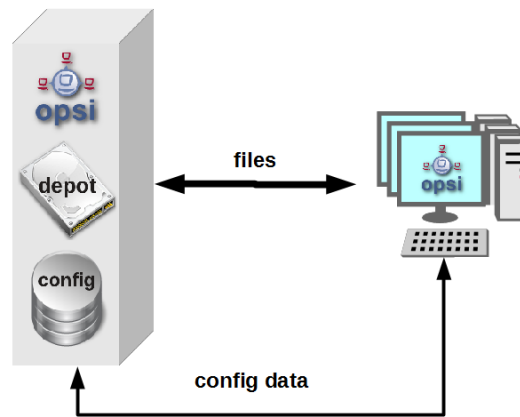


Abbildung 114: Schema eines standalone opsi-servers

Mehrere opsi-Server mit einer zentralen Administration (Multi-Depot Umgebung): Um bei dieser Art der Installation die Checks richtig zu verteilen muss man als erstes Verstehen, wie eine Multi-Depot Umgebung aufgebaut ist:

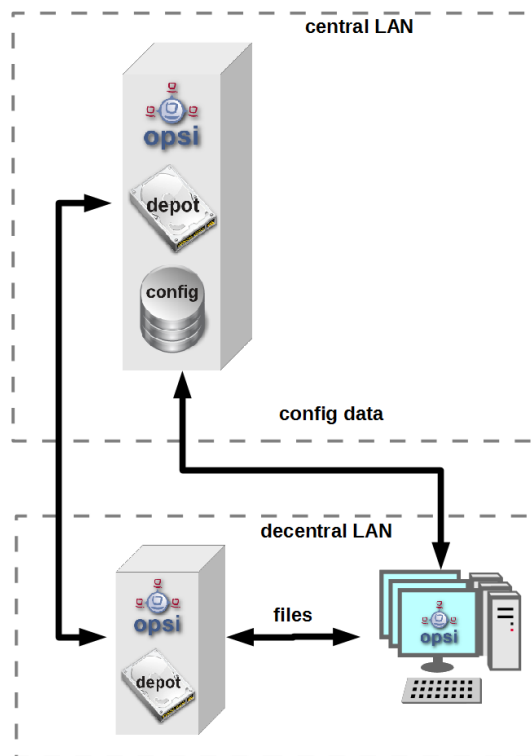


Abbildung 115: Schema einer opsi multidepot installation

Wie im Bild zu erkennen, gibt es nur ein Daten-Backend, welches am opsi-Configserver angesiedelt ist. Jeder Check, der gegen das Backend ausgeführt wird, muss somit zwangsläufig über den opsi-Configserver. Somit werden alle Checks die an die Depotserver gehen, intern an den Configserver weitergeleitet. Deshalb ist es sinnvoller diese Checks direkt gegen den opsi-Configserver aus zu führen. Eine Ausnahme können die aktiven Checks gegen den opsi-client-agent bilden. Wenn zum Beispiel zwischen den Servern eine Firewall aufgestellt ist, die nur den Port 4447 durchlässt, können Clients an der Außenstelle eventuell nicht erreicht werden (Standardport 4441). In solchen Fällen kann es nützlich sein den aktiven Check am Depotserver in der Außenstelle auszuführen.

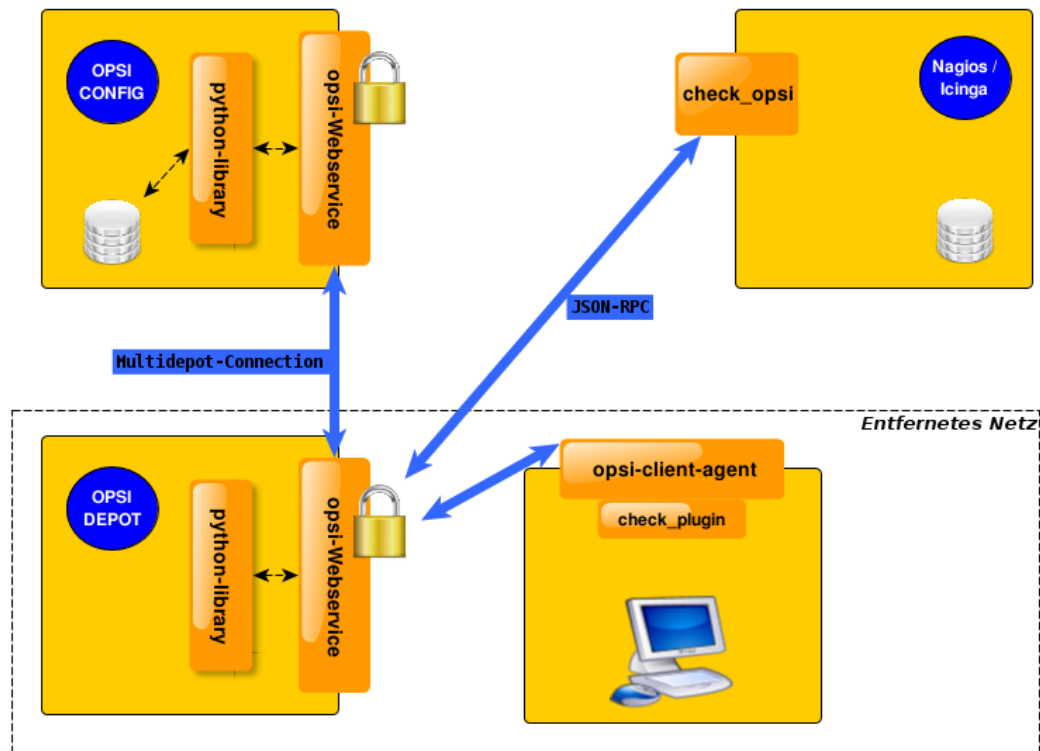


Abbildung 116: Verteilte Checks

opsi-check-plugin

Auf dem Nagios Server gibt es nur ein opsi-check-plugin, welches aber eine große Zahl von Checks unterstützt. Deshalb hat dieses Plugin auch ziemlich viele Optionen. Eine Auflistung dieser Optionen wäre der Erläuterung nicht dienlich, deshalb wird an dieser Stelle darauf verzichtet. Die einzelnen Optionen, die benötigt werden oder möglich sind, werden bei den einzelnen Checks erläutert. Das opsi-check-plugin wird aufgerufen mit dem Befehl `check_opsi`. Eine Übersicht der möglichen Optionen erhält man mit dem Parameter `-h` oder `--help`.

Die folgenden Optionen sind für alle Checks notwendig:

Tabelle 21: Allgemeine Optionen

Optionen	Bezeichnung	Beispiel
<code>-H,--host</code>	opsiServer auf dem gecheckt werden soll	<code>configserver.domain.local</code>
<code>-P,--port</code>	opsi-Webservice Port	4447 (Default)
<code>-u,--username</code>	opsi Monitoring User	monitoring
<code>-p,--password</code>	opsi Monitoring Password	monitoring123
<code>-t,--task</code>	opsi Checkmethode (Case Sensitive)	

Die oben aufgeführten Parameter müssen immer gesetzt werden. Das nachfolgende Kapitel erläutert als erstes, wie man das opsi-check-plugin manuell aufrufen würde. Wie diese über Nagios/Icinga gesetzt werden, wird im Kapitel der Konfiguration erläutert.

Um das check-plugin vom opsi-Nagios-Connector zu installieren, können Sie einfach das opsi-Repository auf dem Nagios-Server eintragen und mit folgendem Befehl das Paket installieren:

```
apt-get install opsi-nagios-plugins
```

Für Redhat/Centos:

```
yum install opsi-nagios-plugins
```

Für OpenSuse/SLES:

```
zypper install opsi-nagios-plugins
```

Das Plugin selbst ist in Python geschrieben und sollte auch auf anderen Distributionen laufen. Dieses Paket basiert auf dem Paket: *nagios-plugins-basic* bzw. *nagios-plugins* und installiert entsprechend das Plugin ins Verzeichnis: `/usr/lib/nagios/plugins`. Die Konfiguration der Nagios-Check-Commands wird nicht automatisch angelegt, da dieses Plugin sehr flexibel einsetzbar ist. Deshalb wird dieser Teil im Kapitel über die Konfiguration etwas später näher erläutert.

Check: opsi-Webservice

Mit diesem Check wird der opsi-Webservice-Prozess überwacht. Dieser Check liefert auch Performancewerte. Deshalb sollte dieser Check auf jedem opsi-Server selbst ausgeführt werden, da jeder opsi-Server seinen eigenen opsiconfd-Prozess hat, der überwacht werden kann bzw. sollte.

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkOpsWebservice
```

Dieser Check liefert in der Regel OK zurück. In folgenden Situationen kann dies Abweichen:

- Critical: Wenn der Webservice ein Problem hat und nicht richtig antwortet. Weiterhin wird dieser Status zurückgemeldet, wenn die CPU-Last des Prozesses 80% überschreitet oder wenn der prozentuale Wert von RPC-Errors 20% erreicht und übersteigt (im Bezug auf die gesamten RPCs).
- Warning: Wenn der Webservice zwar arbeitet, aber Grenzwerte überschreitet: CPU Auslastung höher als 60% aber unter 80% oder wenn der prozentuale Wert der RPC-Errors in Bezug auf die Gesamt-RPC Anzahl höher als 10% aber unter 20% liegt.
- Unknown: Wenn der Webservice gar nicht erreichbar ist.

Info: Die CPU-Werte beziehen sich immer nur auf eine CPU, da ein Prozess auch nur einen Prozessor benutzen kann. Eine Ausnahme bildet hier das Multiprocessing von opsi.

Check: opsi-Webservice pnp4nagios-Template

Für die Performance-Auswertung gibt es ein Template für das pnp4nagios, welches die Werte kombiniert darstellt. Wie man das pnp4nagios installiert, wird hier nicht explizit beschrieben, sondern es wird davon ausgegangen, dass pnp4nagios richtig installiert und konfiguriert wurde. Die notwendige Vorgehensweise kann sich von der hier beschriebenen Lösung unterscheiden, wenn das pnp4nagios mit anderen Pfaden installiert wurde (kann bei selbst kompilierten Installationen vorkommen).

Es werden Standard-Templates verwendet, welche für jeden einzelnen Performancewert ein eigenes Diagramm erstellen. Um das oben genannte Template mit der kombinierten Ansicht zu verwenden, muss man folgendermaßen vorgehen:

Schritt 1: Erstellen Sie unterhalb von: `/etc/pnp4nagios/check_commands` eine Datei mit der Bezeichnung: `check_opsiwebservice.cfg` und folgendem Inhalt:

```
CUSTOM_TEMPLATE = 0
DATATYPE = ABSOLUTE,ABSOLUTE,ABSOLUTE,ABSOLUTE,DERIVE,GAUGE,GAUGE,GAUGE
```

Schritt 2: Legen Sie die Datei `check_opsiwebservice.php` unterhalb von `/usr/share/pnp4nagios/html/templates` ab. Diese Datei können Sie über `svn.opsi.org` auschecken.

```
cd /usr/share/pnp4nagios/html/templates
svn co https://svn.opsi.org/opsi-pnp4nagios-template/trunk/check_opsiwebservice.php
```

Wichtig bei diesen Templates ist, dass die Namen der PHP-Dateien genauso heißen wie der *command_name*, welcher in der Datei `/etc/nagios3/conf.d/opsi/opsicommands.cfg` definiert ist. Stimmen die Bezeichnungen nicht überein, wird ein Standardtemplate von pnp4nagios verwendet.

Sollte diese Anpassung vorgenommen werden, wenn die ersten Checks für den opsi-webservice schon stattgefunden haben, müssen die schon erstellten RRD-Datenbanken erst mal gelöscht werden, da mit diesen Templates auch die Struktur der RRD-Datenbanken neu konfiguriert werden.

In der Regel sind die RRD-Datenbanken unter folgendem Pfad zu finden: `/var/pnp4nagios/perfdata/<host>/` .

Dabei reicht es aus, alle Dateien, die entweder mit `opsi-webservice.rrd` oder mit `opsi-webservice.xml` beginnen, zu löschen. (Vorsicht: Hier werden auch andere RRD-Datenbanken von anderen Checks für diesen Host angelegt, die nicht unbedingt gelöscht werden sollten.).

Damit das Ganze automatisch funktioniert, müssen diese Dateien genauso heißen, wie das `check_command` vom opsi-Webservice. Der Grund dafür liegt in der Arbeitsweise von pnp4nagios. Sollte also die Konfiguration des opsi-Webservice von dieser Dokumentation abweichen, so müssen auch diese Template-Dateien umbenannt werden, da ansonsten pnp4nagios keine richtige Zuordnung treffen kann.

Wurde bis hierhin alles richtig konfiguriert und die Konfigurations-Schritte im Konfigurationskapitel richtig befolgt, sollten die Diagramme wie im folgenden Screenshot automatisch generiert werden:



Check: opsi-check-diskusage

Mit diesem Check werden die opsiiconfd-Ressourcen auf Füllstand überwacht. Die folgende Tabelle zeigt, welche Ressourcen damit gemeint sind.

Tabelle 22: opsi Ressourcen

Ressource	Pfad
/	/usr/share/opsiconfd/static
configured	/usr/lib/configured
depot	/var/lib/opsi/depot
repository	/var/lib/opsi/repository

Bitte auch hier beachten, dass nur die Füllstände der Pfade, die für opsi relevant sind, beim Check berücksichtigt werden. Dieser Check soll keinen allgemeinen DiskUsage-Check ersetzen.

Mit folgendem Befehl kann man alle Ressourcen gleichzeitig abfragen:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkOpsidiskUsage
```

Zusätzlich zu der normalen Checkvariante gibt es die Möglichkeit nur eine Ressource zu checken. Das folgende Beispiel checkt nur nach der Ressource `repository`:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkOpsidiskUsage --resource \
repository
```

Standardmäßig gibt dieser Check OK zurück und gibt den freien Speicherplatz der Ressource oder der Ressourcen zurück. Die Einheit ist dabei Gigabyte. Folgende weitere Zustände sind möglich:

- **WARNING:** Wenn eine oder mehrere Ressourcen 5GB oder weniger freien Speicherplatz haben.
- **CRITICAL:** Wenn eine oder mehrere Ressourcen 1GB oder weniger freien Speicherplatz haben.

Die oben genannten Werte sind die Standard-Thresholds. Diese können durch zusätzliche Angabe von den Optionen `-C` und `-W` bzw. `--critical` und `--warning` selber bestimmt werden. Dabei gibt es zwei mögliche Einheiten: 10G bedeutet mindestens 10 Gigabyte freier Speicherplatz, durch diese Angabe wird der Output auch in dieser Datei ausgegeben. Wenn die Thresholds in Prozent oder ohne Angaben angegeben werden, wird auch der Output in Prozent generiert. Abschließend noch ein Beispiel zur Veranschaulichung:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkOpsidiskUsage --resource \
repository --warning 10% --critical 5%
```

Check: opsi-client-status

Einer von zwei Schwerpunkten des opsi-Nagios-Connectors ist das Überwachen von Software-Rollouts. Dafür wurde dieser Check konzipiert. Er bezieht sich immer auf einen Client innerhalb von opsi.

Der Status der Produkte auf dem Client entscheidet über das Ergebnis des Checks. Auf diese Weise erkennt man schnell, ob eine Produktinstallation auf dem betreffenden Client ansteht oder aktuell ein Problem verursacht hat. Aber nicht der Produktstatus des Clients kann das Checkergebnis beeinflussen, sondern auch wann dieser Client sich das letzte mal am Service gemeldet hat. Wenn der Client länger als 30 Tage nicht am Service war, wird der Check mindestens ein Warning als Ergebnis zurückgeben. Dieser Zeitraum orientiert sich an der Abfolge der Microsoft Patchdays. Wenn ein Client länger als 30 Tagen von der Softwareverteilung abgeschottet ist, sollte man den Client checken. Auf diese Weise kann man auch Clients im System finden, die schon lange inaktiv sind, die man im normalen Betrieb schnell übersieht.

Die Ergebnisse der Tests werden von folgenden zwei Grundregeln bestimmt:

- Der Software rollout Status ist:
 - *OK*
wenn die Software auf dem Client in der selben Produkt- und Paketversion installiert ist, welche auf dem Server liegt und kein *Action Request* gesetzt ist.
 - *Warning*
wenn die auf dem Client installierte Software von der Version auf dem Server abweicht oder ein *Action Request* gesetzt ist.
 - *Critical*
wenn die letzte Aktion für die Software auf dem Client ein *failed* zurückgeliefert hat.
- Die Zeit seit *last seen* liefert:
 - *OK*
wenn der Client vor 30 Tagen oder weniger gesehen wurde.

- *Warning*
wenn der Client vor mehr als 30 Tagen zum letzten Mal gesehen wurde.

Dieser Check kann auf verschiedene Weisen durchgeführt werden; die einfachste Variante:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkClientStatus -c opsiclient.\
domain.local
```

Man kann einzelne Produkte anhand ihre ID von diesem Check ausschließen. Dies würde zum Beispiel so aussehen:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkClientStatus -c opsiclient.\
domain.local -x firefox
```

Das obige Beispiel schließt das Produkt *firefox* aus diesem Check aus, somit würde dieser Check auch ein OK liefern, selbst wenn das Produkt *firefox* bei dem Client auf *failed* steht.

Check: opsi-check-ProductStatus

Der zweite Schwerpunkt des opsi-Nagios-Connectors ist das Überwachen von Software-Rollouts. Mit die wichtigste Aufgabe in einem Software-Verteilungssystem. Sobald eine neue Software, egal ob Standard-Software oder eigene Software mit opsi verwaltet, verteilt und aktuell gehalten wird, muss man den Rollout-Status im Auge behalten.

Das Ergebnis dieses Checks wird durch die folgenden Grundregeln bestimmt:

Der Software Rollout Status ist:

- *OK*
wenn die Software auf dem Client in der selben Produkt- und Paketversion installiert ist, welche auf dem Server liegt und kein *Action Request* gesetzt ist.
- *Warning*
wenn die auf dem Client installierte Software von der Version auf dem Server abweicht oder ein *Action Request* gesetzt ist.
- *Critical*
wenn die letzte Aktion für die Software auf dem Client ein *failed* zurückgeliefert hat.

Durch einige Parameter kann man diesen Check flexibel einsetzen. Um dies besser zu verdeutlichen werden einige Beispielaufufe aufgeführt und erläutert. Im einfachsten Fall ruft man den Check einfach für ein Produkt auf. Dieser wird als *productId* (opsi-interner Name) angegeben.

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkProductStatus -e firefox
```

In einer normalen opsi-Umgebung reicht dieser Aufruf, um den Zustand für das Produkt *firefox* zu überwachen. Die Ausgabe zeigt an, ob alles in Ordnung ist oder wie viele Installationen anstehen (*setup*), wie viele Clients ein Problem mit diesem Produkt haben (*failed*) und wie viele Clients nicht die aktuelle Version installiert haben.

Dies ist zur Übersicht in den meisten Fällen schon ausreichend, wenn man aber nun genau wissen will, auf welchen Clients, welcher Zustand zu diesem Produkt zu diesem Checkergebnis geführt hat, kann man den Befehl im *verbosemode* ausführen:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkProductStatus -e firefox -v
```

Bei einer Multi-Depot Umgebung wird der obige Befehl aber nicht die komplette Umgebung nach diesem Produkt überwachen, sondern nur den Configserver. (Oder exakter: Die Clients die dem depot auf dem config server zugewiesen sind). Wenn man mehrere Depotserver hat, kann man das Depot auch direkt mit angeben:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkProductStatus -e firefox -d \
depotserver.domain.local
```

Der Grund dafür ist, dass jeder Depotserver diese Produkt zur Verteilung haben kann. Dies muss nicht überall die selbe Version sein, auch wenn die `productId` genau die Selbe ist. Aus diesem Grund müssen alle Clients anders beurteilt werden, je nachdem an welchem Depot sie registriert sind. Dies hat zusätzlich noch den Vorteil, dass man diesen Check später im Nagios beim Depotserver ansiedeln kann, was zusätzlich die Übersichtlichkeit erhöht. Wenn man nur ein oder zwei Depotserver hat, kann man auch mit der Angabe von `all` alle Depotserver mit einem Check abdecken oder Komma separiert mehrere Depotserver angeben.

Zusätzlich kann man mit diesem Check auch mit opsi-Gruppen arbeiten. Man kann zum Beispiel eine ganze Produktgruppe mit einem Check abfragen. Wenn man zum Beispiel eine Produktgruppe: `buchhaltung` bildet, kann man mit folgendem Aufruf:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkProductStatus -g buchhaltung
```

Nun werden alle Produkte, die Mitglieder in dieser Produktgruppe sind über diesen Check überwacht. Die Auswertung dieser Gruppe findet immer während des eigentlichen Checks statt; das bedeutet, man kann über opsi diese Gruppe bearbeiten und beeinflusst somit direkt die Check-Parameter ohne das man die Nagios-Konfiguration anpassen muss.

Anmerkung

Gruppen innerhalb von Gruppen werden nicht beachtet und müssen separat angegeben werden.

Auch die Clients, die für diesen Check abgearbeitet werden, können beeinflusst werden. Wie vorher schon erwähnt, wird die Clientliste durch Angabe des Depotserver beeinflusst, zusätzlich können opsi-Hostgruppen angegeben werden, die für diesen Check abgearbeitet werden. Dieser Aufruf sieht zum Beispiel wie folgt aus:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkProductStatus -g buchhaltung -G \
produktivclients
```

Dies würde die Produkte der Produktgruppe `buchhaltung` für alle Clients der Gruppe `produktivclients` überprüfen. Auch bei den Hostgruppen gilt die Regel, dass Untergruppen dieser Gruppe nicht abgearbeitet werden. Für opsi-Productgroups gilt genauso, wie für opsi-Hostgroups, dass mehrere Gruppen Komma separiert angegeben werden können.

Abschließend kann man auch bei diesem Check opsi-Clients ausschließen:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkProductStatus -g buchhaltung -G \
produktivclients -x client.domain.local
```

Check: opsi-check-Depotsync

Gerade in einer Multi-Depot Umgebung ist es wichtig, die Depotserver auf Synchronität zu überwachen. Entscheidend ist bei diesem Check die Software- und Paketversion der installierten Produkte. Manchmal ist ein differenzierter Einsatz der opsi-Produkte auf Depotservern gewünscht, birgt aber die Gefahr, dass bei einem Umzug von einem Client, von einem Depot zum anderen, Inkonsistenzen in der Datenbank entstehen können. Um dieser Problematik entgegen zu wirken, wird empfohlen die opsi-Pakete auf den Depotservern so synchron wie möglich zu halten.

Standardmäßig liefert dieser Check `OK` zurück, sollte eine Differenz festgestellt werden, wird der Status: `WARNING` zurückgegeben. Dieser Check ist ein klassischer Check, der auf dem Configserver ausgeführt werden sollte, da alle Informationen zu diesem Check nur im Backend auf dem opsi-Configserver zu finden sind.

Als nächstes folgen ein paar Anwendungsmöglichkeiten dieses Checks:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkOpsidepotSyncStatus
```

Dies ist die Basis-Variante und äquivalent zu folgendem Aufruf:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkOpsidepotSyncStatus -d all
```

Ohne konkrete Angabe von Depotserver werden die Produkt-Listen aller Depot-Server miteinander verglichen. Um eine bessere Übersichtlichkeit zu schaffen, sollte man diesen Check auf zwei Depotserver reduzieren und lieber auf mehrere Checks verteilen. Dies erreicht man durch direkte Angabe der Depotserver:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkOpsidepotSyncStatus -d \
  configserver.domain.local,depotserver.domain.local
```

Mit diesem Aufruf werden alle Produkte verglichen, die auf beiden Depotservern installiert sind. Sollte ein Produkt auf einem Depotserver gar nicht installiert sein, hat dies keine Auswirkungen auf das Check-Resultat. Dies kann man ändern, indem man bei diesem Check den "strictmode" Schalter setzt:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkOpsidepotSyncStatus -d \
  configserver.domain.local,depotserver.domain.local --strictmode
```

Nun werden auch Produkte angezeigt, die auf einem Depotserver nicht installiert sind. Um ein bestimmtes Produkt oder bestimmte Produkte nicht mit zu checken, weil man zum Beispiel will dass diese Produkte in verschiedenen Versionen eingesetzt werden, kann man diese Produkte von diesem Check ausschließen:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkOpsidepotSyncStatus -d \
  configserver.domain.local,depotserver.domain.local --strictmode -x firefox,thunderbird
```

Dieser Check würde auch dann ein OK zurückgeben, wenn das *firefox* und das *thunderbird*-Paket nicht überall synchron eingesetzt werden.

Ein weitere Einsatzmöglichkeit wäre, dass nur eine Auswahl von Produkten auf Synchronität überwacht werden können. Dies kann man durch:

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkOpsidepotSyncStatus -d \
  configserver.domain.local,depotserver.domain.local --strictmode -e firefox,thunderbird
```

So werden nur *firefox* und *thunderbird* Produkte auf Synchronität überwacht. Bei diesem Check sollte der `strictmode` gesetzt sein, damit man auch erkennt, wenn die gewünschten Produkte auf Depotservern nicht installiert sind.

Check: Plugin über Opsiclientd checken

Dieser Check führt ein Check-Plugin auf dem Client direkt aus und fängt die Ausgabe ein.

Diese Erweiterung soll keinen Ersatz für einen richtigen Nagios-Agent bieten, sondern eine Alternative. Man kann diese Erweiterung einsetzen, wenn man Plugins auf dem opsi-Client checken will. Die eingesetzten Plugins müssen den sogenannten [Nagios plug-in development guidelines](#) entsprechen.

Um ein Plugin ausführen zu können, muss man das Plugin erst einmal auf den Clients verteilen. Dies sollte man über ein opsi-Paket lösen. Der Ablageort für die Plugins auf dem Client ist im ersten momentan egal, da man den Pfad beim Checken mit angeben muss. Allerdings sollte man die Plugins nicht einzeln verteilen, sondern in einem Verzeichnis zusammenführen, damit das Aktualisieren und Pflegen der Plugins einfacher wird. Weiterhin sollte man auch Sicherheitstechnisch im Hinterkopf behalten, dass die Plugins im Systemkontext des opsi-clientd-Services aufgerufen werden. Normale Anwender sollten auf dieses Verzeichnis keinen Zugang haben.

Es gibt diverse Plugins, die es schon vorgefertigt im Internet runter zu laden gibt. Eine mögliche Anlaufstelle ist [Nagios Exchange](#).

Im Folgenden wird davon ausgegangen, dass unter `C:\opsi.org\nagiosplugins\` das Plugin `check_win_disk.exe` vom Paket [nagioscol](#) abgelegt wurde.

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkPluginOnClient --plugin "C:\\\
  opsi.org\nagiosplugincol\check_win_disk.exe C:" -c client.domain.local
```

Dieser Aufruf checkt auf dem Client `client.domain.local` das Plugin `check_win_disk.exe` und übergibt diesem den Parameter `C:`. Dies bedeutet, dass das Laufwerk `C` auf dem Client gecheckt wird. Die Ausgabe und der Rückgabewert dieses Plugins wird direkt vollständig ausgewertet und diese Werte können in Nagios unmittelbar weiter verarbeitet werden.

Ein besonderes Feature ist das Beibehalten von Zuständen. Diese Implementation ist aus der Problemstellung entstanden, dass Clients nicht wie Server durchlaufen, sondern in der Regel nur einen bestimmten Zeitraum eingeschaltet sind. Man kann den Check auf Nagios-Seite zwar mit sogenannten `Timeperiods` eingrenzen, aber in der Praxis ist

so ein Vorgehen nicht praktikabel, da man zum Beispiel auch bei Urlaub von Anwendern flexibel reagieren muss. Dies würde eine ständige Konfigurationsarbeit nach sich ziehen. Wenn man darauf verzichtet, wird der Status ständig geändert, auch wenn ein aufgetretenes Problem noch gar nicht gelöst ist. Deshalb kann man den letzten bekannten Status an opsi übergeben. Sollte der Client nicht erreichbar sein, wird dieser letzte bekannte Status zurückgegeben. Ein *Critical*-Zustand bleibt beispielsweise in diesem Falle auch auf *Critical* stehen und wechselt nicht auf *Unknown*, was rein logisch aber richtig wäre.

Um dieses Feature später mit Nagios zu verwenden, kann man die Nagios-Makros: `$$SERVICESTATEID$` und `$$SERVICEOUTPUT$` nutzen.

```
check_opsi -H configserver.domain.local -P 4447 -u monitoring -p monitoring123 -t checkPluginOnClient --plugin "C:\\opsi.org\\nagiosplugincol\\check_win_disk.exe C:" -c client.domain.local -s $$SERVICESTATEID$ -o $$SERVICEOUTPUT$
```

9.11.5 opsi Monitoring Konfiguration

Dieses Kapitel widmet sich der Konfiguration der Schnittstelle von opsi und dem Nagios-Server. Die Konfigurationen in diesem Kapitel, besonders die auf dem Nagios-Server, sollen als Empfehlungen gelten, sind aber nicht die einzigen Lösungen. Hier wird nur die Konfiguration mit einem Nagios-Server beschrieben. Mit einem Icinga-Server sollte, mit Ausnahme von ein paar Pfaden, die Konfiguration ziemlich genauso funktionieren. Andere Derivate auf Nagios Basis sollten funktionieren, wurden aber nicht getestet.

Tipp

Die Konfigurationsdateien aus diesem Kapitel sind Bestandteil des opsi-nagios-connector-utils svn-Repository. Um die Beispiel-Konfigurationsdateien direkt zu beziehen können Sie auf dieses Repository per Browser zugreifen:

```
https://svn.opsi.org/listing.php?repname=opsi-nagios-connector-utils
```

oder direkt per svn ein Checkout ausführen:

```
svn co https://svn.opsi.org/opsi-nagios-connector-utils
```

opsi Monitoring User

In der Regel wird im Monitoring-Bereich viel mit IP-Freischaltungen als Sicherheit gearbeitet. Da aber dieser Mechanismus nicht wirklich einen Schutz bietet, wurde beim opsi-Nagios-Connector darauf verzichtet. Aus diesem Grund wird das Ganze per Benutzer und Passwort geschützt. Diesen User als opsi-admin einzurichten, würde aber auch hier zu viele Rechte freischalten, da dieser User nur für diese Schnittstelle von Nöten ist und auch die Benutzbarkeit auf diesen Bereich eingeschränkt werden soll, wird der User nur intern in opsi eingerichtet. Folgender Befehl legt den User an:

```
opsi-admin -d method user_setCredentials monitoring monitoring123
```

Dieser Befehl legt den User: monitoring mit dem Passwort monitoring123 an. Der User wird in der `/etc/opsi/passwd` angelegt und ist auch kein User, mit dem man sich an der Shell anmelden könnte.

Bei einer Multi-Depot Umgebung muss man diesen User nur auf dem Configserver erzeugen.

Beim Nagios-Server kann man dieses Passwort vor den CGI-Skripten maskieren, indem man einen Eintrag in der `/etc/nagios3/resource.cfg` vornimmt. Dieser sieht zum Beispiel so aus:

```
$$USER2$=monitoring
$$USER3$=monitoring123
```

Die Zahl hinter `$$USER` kann variieren. Wenn diese Datei vorher nicht genutzt wurde, sollte in der Regel nur das `$$USER1$` belegt sein. Diese Konfiguration dient als Grundlage für die weiteren Konfigurationen.

opsi Nagios-Connector Konfigurationsverzeichnis

Aus Gründen der Übersichtlichkeit empfiehlt es sich für die Konfigurationsdateien des opsi-Nagios-Connectors ein eigenes Verzeichnis anzulegen. Erzeugen Sie dazu unterhalb von: `/etc/nagios3/conf.d` ein Verzeichnis `opsi`. Dies macht die Konfiguration später übersichtlicher, da alle Konfigurationen, die den opsi-Nagios-Connector betreffen, gebündelt auf dem Server abgelegt sind.

Zu den Konfigurationsdateien in diesem Verzeichnis gehören:

- Nagios Template: `opsitemplates.cfg`
- Hostgroups: `opsihostgroups.cfg`
- Server Hosts: `<full name of the server>.cfg`
- Kommandos: `opsicheckcommands.cfg`
- Kontakte: `opsicontacts.cfg`
- Services: `opsiservices.cfg`

Um die Übersichtlichkeit noch weiter zu erhöhen sollten Sie unterhalb von `/etc/nagios3/conf.d/opsi` noch ein Verzeichnis `clients` anlegen, das im Weiteren dazu dient, die Konfigurationsdateien für die Clients aufzunehmen.

Nagios Template: `opsitemplates.cfg`

Es gibt diverse Templates für diverse Objekte im Nagios. Dies ist eine Standard-Nagios Funktionalität, die hier nicht näher beschrieben wird. Diese Funktionalität kann man sich für opsi zu nutze machen, um sich später bei der Konfiguration die Arbeit zu vereinfachen.

Da die meisten Checks auf dem Configserver ausgeführt werden, sollte man sich als erstes ein Template für die opsi-Server und ein Template für die opsi-Clients schreiben. Da es in einer Multi-Depot Umgebung nur einen Configserver geben kann, kann man direkt diesen ins Template übernehmen. Dies erreicht man am einfachsten über eine Custom-Variable. Diese erkennt man daran, dass sie mit einem `_` beginnen. Es wird in das Template für den opsi-Server und die opsi-Clients folgendes zusätzlich eingetragen:

```
_configserver      configserver.domain.local
_configserverurl   4447
```

Auf diese beiden *custom Variablen* kann man später einfach mit dem Nagios-Makro: `$_HOSTCONFIGSERVER$` und `$_HOSTCONFIGSERVERPORT$`, verweisen. HOST muss vorher angegeben werden, da diese Custom-Variablen in einer Hostdefinition vorgenommen wurden. Weitere Informationen zu Custom-Variablen entnehmen Sie bitte der Nagios-Dokumentation. Da diese beiden Konfigurationen im Template vorgenommen werden müssen, gelten sie für jede Hostdefinition, die später von diesen Templates erben. Aber dazu später mehr.

Um nun die Templates anzulegen, sollte man unterhalb von: `/etc/nagios3/conf.d` als erstes ein Verzeichnis `opsi` erstellen. Dies macht die Konfiguration später übersichtlicher, da alle Konfigurationen, die den opsi-Nagios-Connector betreffen, gebündelt auf dem Server abgelegt sind. In diesem Verzeichnis erstellt man die Datei und benennt sie direkt so, dass man später erkennt, was genau hier konfiguriert werden soll: `opsitemplates.cfg`.

In dieser Datei können verschiedene Templates definiert werden. Die Templatedefinition richtet sich dabei nach folgendem Muster, bei dem zur besseren Verständlichkeit die Kommentare zu den einzelnen Einstellungen nicht gelöscht wurden:

```
define host{
    name                opsihost-tmp      ; The name of this host template
    notifications_enabled 1                ; Host notifications are enabled
    event_handler_enabled 1                ; Host event handler is enabled
    flap_detection_enabled 1               ; Flap detection is enabled
    failure_prediction_enabled 1           ; Failure prediction is enabled
    process_perf_data     0                ; Process performance data
    retain_status_information 1            ; Retain status information across program restarts
    retain_nonstatus_information 1         ; Retain non-status information across program restarts
}
```

```

        max_check_attempts      10
        notification_interval    0
        notification_period      24x7
        notification_options     d,u,r
        contact_groups           admins
register      0      ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL HOST, JUST A TEMPLATE!
icon_image   opsi/opsi-client.png
}

```

- Optional kann durch die Option `icon_image` ein Image gesetzt werden, dieser muss relativ zum Pfad: `/usr/share/nagios3/htdocs/images/logos/` angegeben werden.
- Optional kann auch eine eigene `contact_group` angegeben werden, die allerdings als Contact-Object z.B. in der `opsicontacts.cfg` angelegt sein muss.

Wir empfehlen Templates für folgende Objekte anzulegen:

- opsi server
- opsi client
- opsi service
- sowie 2 templates für pnp4nagios (host-pnp / srv-pnp)

Zunächst das Beispiel des opsi-Server-Templates:

```

define host{
    name                opsi-server-tmpl
    notifications_enabled 1
    event_handler_enabled 1
    flap_detection_enabled 1
    failure_prediction_enabled 1
    process_perf_data 1
    retain_status_information 1
    retain_nonstatus_information 1
    check_command        check-host-alive
    max_check_attempts   10
    notification_interval 0
    notification_period   24x7
    notification_options  d,u,r
    contact_groups        admins,opsiadmins
    _configserver         configserver.domain.local
    _configserverport     4447
    register              0
    icon_image            opsi/opsi-client.png
}

```

Hier muss natürlich noch `configserver.domain.local` an die lokalen Gegebenheiten angepasst werden. Auch die `contact_groups` bedürfen evtl. der Anpassung.

Als nächster Teil der Datei `opsitemplates.cfg` das Template für die Clients:

```

define host{
    name                opsi-client-tmpl
    notifications_enabled 1
    event_handler_enabled 1
    flap_detection_enabled 1
    failure_prediction_enabled 1
    process_perf_data 1
    retain_status_information 1
    retain_nonstatus_information 1
    max_check_attempts   10
    notification_interval 0
}

```

```

notification_period      24x7
notification_options     d,u,r
contact_groups           admins,opsiadmins
_configserver             configserver.domain.local
_configserverport        4447
register                  0
icon_image                opsi/opsi-client.png
}

```

Da die Clients in der Regel nicht durchlaufen, sollte die Option: "check command check-host-alive" nicht gesetzt werden. Somit werden die Clients als Pending angezeigt und nicht als Offline.

Hier muss natürlich noch *configserver.domain.local* an die lokalen Gegebenheiten angepasst werden. Auch die *contact_groups* bedürfen evtl. der Anpassung.

Als nächster Teil der Datei *opsitemplates.cfg* das Template für die opsi-services:

```

define service{
    name                opsi-service-tmpl
    active_checks_enabled 1
    passive_checks_enabled 1
    parallelize_check     1
    obsess_over_service   1
    check_freshness       0
    notifications_enabled 1
    event_handler_enabled 1
    flap_detection_enabled 1
    failure_prediction_enabled 1
    process_perf_data     1
    retain_status_information 1
    retain_nonstatus_information 1
        notification_interval      0
        is_volatile                 0
        check_period                 24x7
        normal_check_interval        5
        retry_check_interval         1
        max_check_attempts           4
        notification_period          24x7
        notification_options         w,u,c,r
        contact_groups               admins,opsiadmins
    register              0
}

```

Wenn *pnnp4nagios* für die grafische Darstellung der Performancedaten für den opsi-Webservice eingesetzt werden, sollten noch die folgenden zwei Objekte als Template in der Datei *opsitemplates.cfg* angelegt werden:

```

define host {
    name        host-pnp
    action_url  /pnp4nagios/index.php/graph?host=$HOSTNAME&&srv=_HOST_
    register    0
}

define service {
    name        srv-pnp
    action_url  /pnp4nagios/index.php/graph?host=$HOSTNAME&&srv=$SERVICEDESC$
    register    0
}

```

opsi Hostgroup: opsihostgroups.cfg

Als nächstes sollten folgende Hostgruppen erstellt werden. Dies dient zum einen der besseren Übersicht auf der Weboberfläche und hilft beim Konfigurieren der Services. Dafür sollte eine Datei Namens *opsihostgroups.cfg* mit folgendem Inhalt erstellt werden:

```

define hostgroup {
    hostgroup_name opsi-clients
    alias          OPSI-Clients
}

define hostgroup {
    hostgroup_name opsi-server
    alias          OPSI-Server
    members        configserver.domain.local, depotserver.domain.local
}

```

opsi Server: <full name of the server>.cfg

Als nächstes sollten die opsi-Server konfiguriert werden. Dies kann jeweils in einer eigenen Datei wie zum Beispiel `configserver.domain.local.cfg` oder eine Datei mit allen opsi-Hosts wie `opsihost.cfg` erfolgen. Der Inhalt sollte für opsi-Server folgendermaßen aussehen:

```

define host{
    use                opsi-server-tmpl
    host_name          configserver.domain.local
    hostgroups         opsi-server
    alias              opsi Configserver
    address            configserver.domain.local
}

define host{
    use                opsi-server-tmpl
    host_name          depotserver.domain.local
    hostgroups         opsi-server
    alias              opsi Depotserver
    address            depotserver.domain.local
}

```

Folgende Erläuterungen zu den Werten: * *use* bezeichnet, welches Template benutzt wird. * *hostgroups* gibt an, welcher Hostgruppe der Server angehört.

opsi Clients: clients/<full name of the client>.cfg

Die opsi-Clients sollten mindestens folgendermaßen definiert werden:

```

define host{
    use                opsi-client-tmpl
    host_name          client.domain.local
    hostgroups         opsi-clients
    alias              opsi client
    address            client.domain.local
    _depotid           depotserver.domain.local
}

```

Die Clientkonfiguration bedient sich einer Sonderfunktion von Nagios. Die Option `_depotid` ist eine sogenannte benutzerdefinierte Variable, die als Makro `$_HOSTDEPOTID$` benutzt werden kann. Die Verwendung ist optional und wird verwendet, wenn die Ausführung eines direkten Checks nicht vom config server, sondern vom hier definierten Depotserver aus startet.

Um die Erstellung von vielen Client Konfigurationsdateien zu vereinfachen, können diese auf dem opsi-Config-Server mit folgendem Skript erstellt werden.

```

#!/usr/bin/env python

from OPSI.Backend.BackendManager import *

template = '''

```



```

define host {
    use                opsi-client-tmpl
    host_name          %hostId%
    hostgroups         opsi-clients
    alias              %hostId%
    address            %hostId%
}
'''

backend = BackendManager(
    dispatchConfigFile = u'/etc/opsi/backendManager/dispatch.conf',
    backendConfigDir   = u'/etc/opsi/backends',
    extensionConfigDir = u'/etc/opsi/backendManager/extend.d',
)

hosts = backend.host_getObjects(type="OpsIClient")

for host in hosts:
    filename = "%s.cfg" % host.id
    entry = template.replace("%hostId%", host.id)
    f = open(filename, 'w')
    f.write(entry)
    f.close()

```

opsi Check-Kommandos Konfiguration: opsicommands.cfg

Die oben beschriebenen Check-Kommandos müssen nun mit den bisherigen Kommandos konfiguriert werden. Dafür sollte eine Datei mit dem Namen `opsicommands.cfg` erstellt werden. Hier wird eine Beispielkonfiguration angegeben; je nach dem welche Funktionen Sie auf welche Weise verwenden wollen, müssen Sie diese Einstellungen nach eigenem Ermessen modifizieren.

Als erstes wird der Aufbau anhand eines Beispiels erläutert:

```

define command{
    command_name    check_opsi_clientstatus
    command_line    $USER1$/check_opsi -H $_HOSTCONFIGSERVER$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$ -p \
    $USER3$ -t checkClientStatus -c $HOSTADDRESS$
}

```

Der `command_name` dient zur Referenzierung der weiteren Konfiguration. In der Option `command_line` werden die Konfigurationen und das Check-Kommando als erstes vereint.

Nach diesem Prinzip baut man nun die ganze Datei `opsicommands.cfg` auf:

```

define command {
    command_name    check_opsiwebservice
    command_line    /usr/lib/nagios/plugins/check_opsi -H $HOSTADDRESS$ -P 4447 -u $USER2$ -p $USER3$ -t \
    checkOpsiWebservice
}
define command {
    command_name    check_opsidiskusage
    command_line    /usr/lib/nagios/plugins/check_opsi -H $HOSTADDRESS$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$ -p \
    $USER3$ -t checkOpsiDiskUsage
}
define command {
    command_name    check_opsiclientstatus
    command_line    /usr/lib/nagios/plugins/check_opsi -H $_HOSTCONFIGSERVER$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$\
    -p $USER3$ -t checkClientStatus -c $HOSTADDRESS$
}
define command {
    command_name    check_opsiproductstatus
    command_line    /usr/lib/nagios/plugins/check_opsi -H $_HOSTCONFIGSERVER$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$\
    -p $USER3$ -t checkProductStatus -e $ARG1$ -d $HOSTADDRESS$ -v
}

```

```

define command {
    command_name    check_opsiproductStatus_withGroups
    command_line    /usr/lib/nagios/plugins/check_opsi -H $_HOSTCONFIGSERVER$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$\
    -p $USER3$ -t checkProductStatus -g $ARG1$ -G $ARG2$ -d "all"
}
define command {
    command_name    check_opsiproductStatus_withGroups_long
    command_line    /usr/lib/nagios/plugins/check_opsi -H $_HOSTCONFIGSERVER$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$\
    -p $USER3$ -t checkProductStatus -g $ARG1$ -G $ARG2$ -v -d "all"
}
define command {
    command_name    check_opsidepotsync
    command_line    /usr/lib/nagios/plugins/check_opsi -H $_HOSTCONFIGSERVER$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$\
    -p $USER3$ -t checkDepotSyncStatus -d $ARG1$
}
define command {
    command_name    check_opsidepotsync_long
    command_line    /usr/lib/nagios/plugins/check_opsi -H $_HOSTCONFIGSERVER$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$\
    -p $USER3$ -t checkDepotSyncStatus -d $ARG1$ -v
}
define command {
    command_name    check_opsidepotsync_strict
    command_line    /usr/lib/nagios/plugins/check_opsi -H $_HOSTCONFIGSERVER$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$\
    -p $USER3$ -t checkDepotSyncStatus -d $ARG1$ --strict
}
define command {
    command_name    check_opsidepotsync_strict_long
    command_line    /usr/lib/nagios/plugins/check_opsi -H $_HOSTCONFIGSERVER$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$\
    -p $USER3$ -t checkDepotSyncStatus -d $ARG1$ --strict -v
}
define command {
    command_name    check_opsipluginon_client
    command_line    /usr/lib/nagios/plugins/check_opsi -H $_HOSTCONFIGSERVER$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$\
    -p $USER3$ -t checkPluginOnClient -c $HOSTADDRESS$ --plugin $ARG1$
}
define command {
    command_name    check_opsipluginon_client_with_states
    command_line    /usr/lib/nagios/plugins/check_opsi -H $_HOSTCONFIGSERVER$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$\
    -p $USER3$ -t checkPluginOnClient -c $HOSTADDRESS$ --plugin $ARG1$ -s $SERVICESTATEID$ -o "$SERVICEOUTPUT$"
}
define command {
    command_name    check_opsipluginon_client_from_depot
    command_line    /usr/lib/nagios/plugins/check_opsi -H $_HOSTDEPOTID$ -P $_HOSTCONFIGSERVERPORT$ -u $USER2$ -p \
    $USER3$ -t checkPluginOnClient -c $HOSTADDRESS$ --plugin $ARG1$
}

```

Kontakte: opsicontacts.cfg

```

define contact{
    contact_name    adminuser
    alias           Opsi
    service_notification_period    24x7
    host_notification_period    24x7
    service_notification_options    w,u,c,r
    host_notification_options    d,r
    service_notification_commands    notify-service-by-email
    host_notification_commands    notify-host-by-email
    email           root@localhost
}
define contactgroup{
    contactgroup_name    opsiadmins
    alias           Opsi Administrators
    members           adminuser
}

```

Hier müssen Sie natürlich *adminuser* in einen bzw. mehrere reale User abändern.

Services: opsiservices.cfg

In der Konfiguration der *Services* wird nun endlich angegeben, was der Nagios-Server monitoren und anzeigen soll. Dabei wird nun auf die bisher definierten Templates, commands und hostgroups bzw. hosts zugegriffen.

Zunächst der Teil der Services, die die tatsächlichen Zustände der Server betreffen. Hierbei wird beim Check auf den Depotsync gegen alle bekannten Services (*all*) geprüft.

```
#OPSI-Services
define service{
    use                opsi-service-tmpl, srv-pnp
    hostgroup_name     opsi-server
    service_description opsi-webservice
    check_command       check_opsiwebservice
    check_interval     1
}

define service{
    use                opsi-service-tmpl
    hostgroup_name     opsi-server
    service_description opsi-diskusage
    check_command       check_opsidiskusage
    check_interval     1
}

define service{
    use                opsi-service-tmpl
    hostgroup_name     opsi-server
    service_description opsi-depotsyncstatus-longoutput
    check_command       check_opsidepotsync_long!all
    check_interval     10
}

define service{
    use                opsi-service-tmpl
    hostgroup_name     opsi-server
    service_description opsi-depotsyncstatus-strict-longoutput
    check_command       check_opsidepotsync_strict_long!all
    check_interval     10
}
```

Nun der Teil der das Softwarerollout überwacht. Dabei wird in einem Check auf das opsi-Produkt *opsi-client-agent* verwiesen und zwei andere Checks verwenden hier eine opsi-Produktgruppe *opsiessentials* sowie eine opsi-Clientgruppe *productiveclients*.

```
define service{
    use                opsi-service-tmpl
    hostgroup_name     opsi-clients
    service_description opsi-clientstatus
    check_command       check_opsiclientstatus
    check_interval     10
}

define service{
    use                opsi-service-tmpl
    hostgroup_name     opsi-server
    service_description opsi-productstatus-opsiclientagent
    check_command       check_opsiproductstatus!opsi-client-agent
    check_interval     10
}

define service{
    use                opsi-service-tmpl
    hostgroup_name     opsi-server
    service_description opsi-productstatus-opsiessentials-group
    check_command       check_opsiproductStatus_withGroups!opsiessentials!productiveclients
    check_interval     10
}

define service{
    use                opsi-service-tmpl
    hostgroup_name     opsi-server
    service_description opsi-productstatus-opsiessentials-group-longoutput
}
```

```

check_command      check_opsiproductStatus_withGroups_long!opsiessentials!productiveclients
check_interval     10
}

```

Im dritten und letzten Teil werden die direkten Checks für die Clients definiert. Diese Checks richten sich in diesem Beispiel nicht an hostgroups, sondern einzelne hosts (*client.domain.local, depotclient.domain.local*).

- *opsi-direct-checkpluginonclient*
Liefert einen normalen direkten Check vom Client und ein unknown, wenn der Client ausgeschaltet ist. Dabei wird versucht den Client direkt vom Configserver aus zu erreichen.
- *opsi-direct-checkpluginonclient-with-servicestate*
verhält sich analog zu *opsi-direct-checkpluginonclient* doch wenn der Client ausgeschaltet ist, liefert er das letzte erfolgreiche Checkergebnis.
- *opsi-direct-checkpluginonclient-from-depot*
verhält sich analog zu *opsi-direct-checkpluginonclient* nur wird versucht, den Client von der, in der Client Konfiguration angegeben, *_depotid* zu erreichen.

```

define service{
    use                opsi-service-tmpl
    host_name          client.domain.local,depotclient.domain.local
    service_description opsi-direct-checkpluginonclient
    check_command      check_opsipluginon_client!"C:\\opsi.org\\nagiosplugins\\check_memory.exe"
    check_interval     10
}
define service{
    use                opsi-service-tmpl
    host_name          client.domain.local
    service_description opsi-direct-checkpluginonclient-with-servicestate
    check_command      check_opsipluginon_client_with_states!"C:\\opsi.org\\nagiosplugins\\
check_memory.exe"
    check_interval     10
}
define service{
    use                opsi-service-tmpl
    host_name          depotclient.domain.local
    service_description opsi-direct-checkpluginonclient-from-depot
    check_command      check_opsipluginon_client_from_depot!"C:\\opsi.org\\nagiosplugins\\check_memory\
.exe"
    check_interval     10
}

```

9.12 opsi-clonezilla (frei)

9.12.1 Vorbedingungen für die opsi Erweiterung *opsi-clonezilla*

Technische Voraussetzungen sind opsi 4.0.3 mit den Paketständen:

Tabelle 23: Benötigte Pakete

opsi-Paket	Version
opsi-linux-bootimage	>= 20130207-1

bzw. opsi 4.0.5 mit den Paketständen:

Tabelle 24: Benötigte Pakete

opsi-Paket	Version
opsi-linux-bootimage	>= 20140805-1
opsi-clonezilla	>= 4.0.5-1

Achtung

Für das Produkt `opsi-clonezilla` muß der share `opsi_images` für `pcpatch` beschreibbar sein. Prüfen Sie Ihre Samba Konfiguration.

Unterstützung für UEFI Maschinen erst ab opsi 4.0.7

Stellen Sie das Property `imageshare` auf einen share, welcher vom user `pcpatch` mit dem dem opsi-server bekannten Passwort gemountet werden kann. Das Format für den share ist dabei `//server/share` (Beachten Sie die Verwendung von slashes statt backslashes). Dieser share ist üblicherweise der share `opsi_images` des opsi servers.

9.12.2 Einführung

Neben der Paketbasierten (`unattended`) installation hatte opsi in der Vergangenheit nur eine rudimentäre Unterstützung für Image basierte Installationen. Mit der Integration der Technik des Open Source Produktes `clonezilla` stellen wir nun eine umfangreiche und flexible Lösung zum Umgang mit Partitions- und Plattenimages vor.

9.12.3 Konzept

Wir haben die `clonezilla` scripte mit dem `opsi-linux-bootimage` kombiniert. Dadurch ergeben sich folgende Vorteile:

- Integration in die Steuerung durch opsi
- Automatischer mount des shares zur Ablage der images
- Möglichkeit der Automatisierung der Abläufe

9.12.4 Interaktive Abläufe

Per default läuft `opsi-clonezilla` im interaktiven Modus. Dieser erlaubt es einfach und bequem die Parameter für einzelne Aktionen zu bestimmen und kann aber auch als Basis für die Automatisierung dienen.

- Stellen Sie das Property `runcommand` auf `ocs-live`. Dies ist der Interaktive Modus von `clonezilla`.
- Starten Sie das Netbootprodukt.
- In der ersten Maske werden Sie gefragt ob nach `/home/partimg` etwas gemountet werden soll. Wählen Sie `Skip` da dies vom bootimage schon erledigt worden ist.

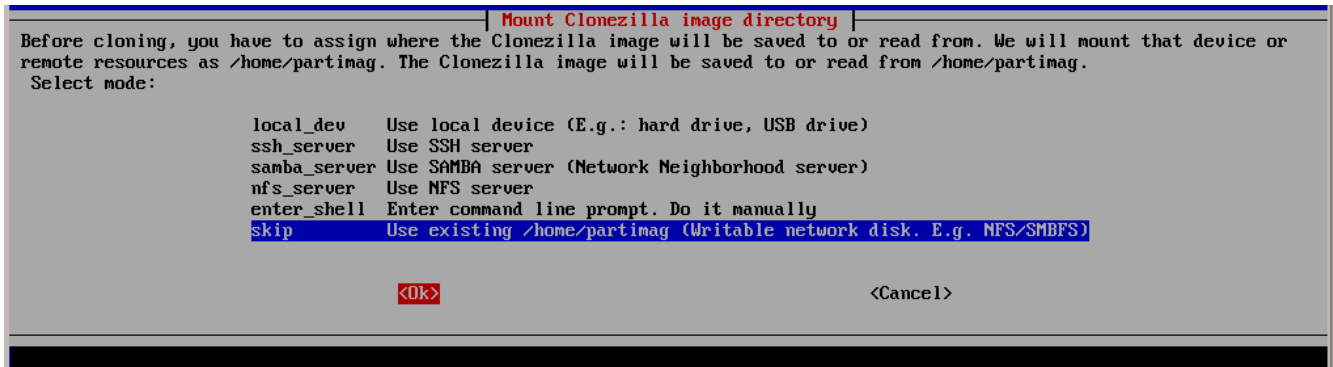


Abbildung 117: Skip: Der über das Property 'imageshare' angegebene Ziel- / Quell-share wird vom opsi-bootimage nach /home/partimag gemountet.

Die gemounteten Partitionen werden angezeigt:

```

csroot device is skip
existing setting is:
*****.
filesystem      Size  Used Avail Use% Mounted on
rootfs          0     0     0  -  /
proc            0     0     0  -  /proc
sys            0     0     0  -  /sys
one            0     0     0  -  /sys/kernel/debug
one            0     0     0  -  /sys/kernel/security
one            506M  128K  506M  1%  /dev
one            0     0     0  -  /dev/pts
one            506M  0     506M  0%  /dev/shm
one            506M  28K  506M  1%  /var/run
one            506M  0     506M  0%  /var/lock
one            506M  0     506M  0%  /lib/init/rw
/sepiolina/opsi_depot
                868G  356G  469G  44%  /mnt/opsi
/sepiolina/opsi_images
                868G  356G  469G  44%  /home/partimag
*****.
press "Enter" to continue.....

```

Abbildung 118: Gemountete Partitionen.

In der Abfrage **Expert** oder **Beginner** bestimmen Sie ob Sie im folgenden die Defaultparameter ungefragt übernehmen (**Beginner**) oder diese abändern können (**Expert**).

Ab opsi 4.0.5 können Sie auch den **Beginner** Modus wählen.

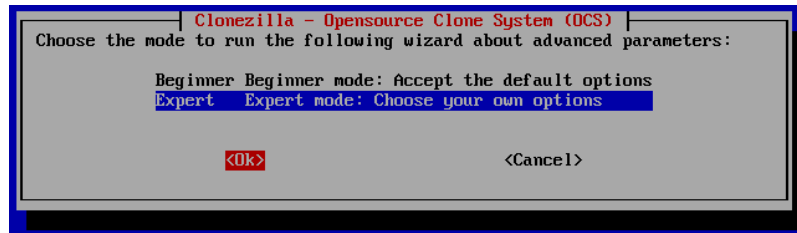


Abbildung 119: Experte oder lieber einfach ?

Nun können Sie wählen welche Operation Sie durchführen wollen. Hier werden die folgenden Operationen behandelt:

- save disk
- save partition
- restore disk
- restore partition

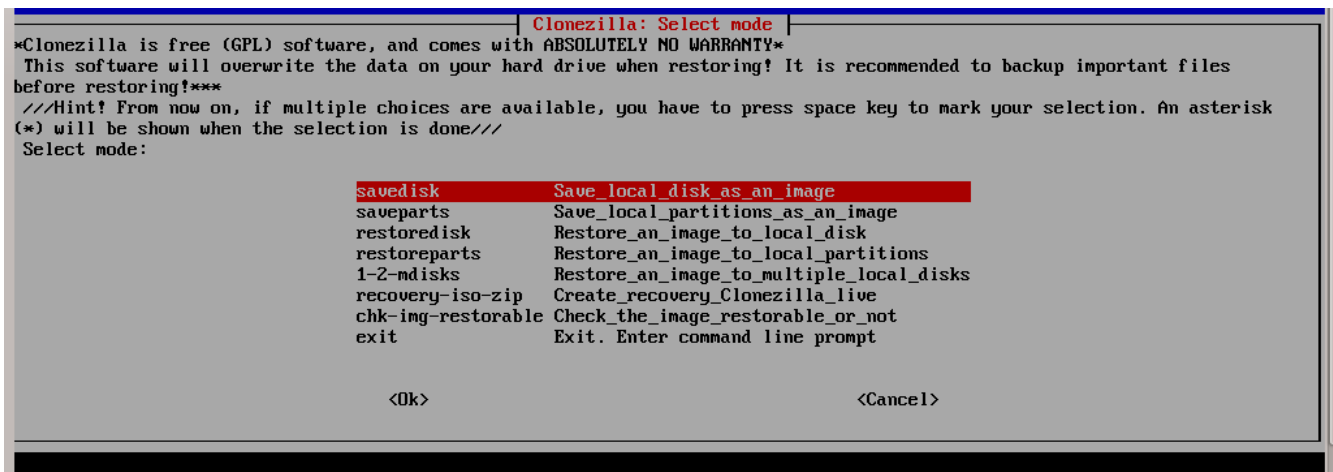


Abbildung 120: Operation wählen.

Interaktives savedisk im expert mode

Hier beispielhaft anhand der savedisk Operation die zusätzlichen Masken des Expertmodes.

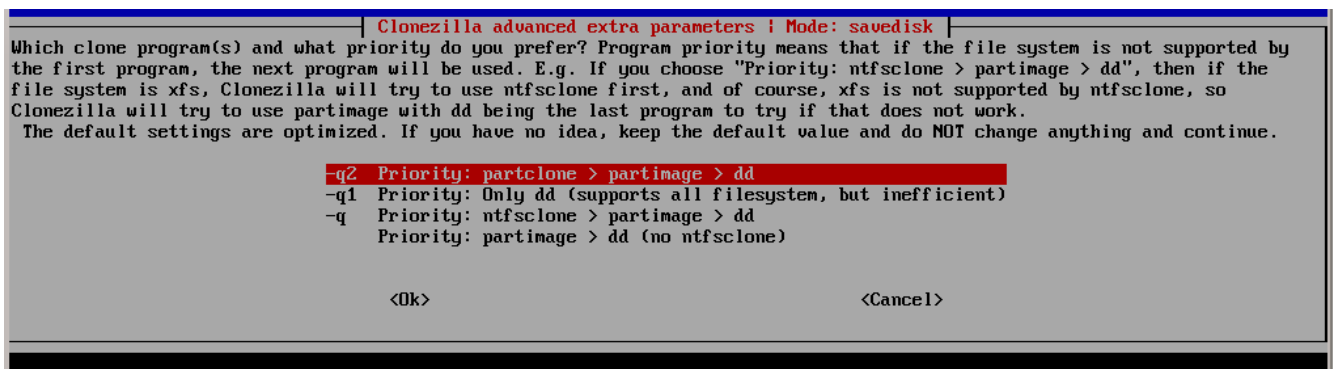


Abbildung 121: Wahl der Werkzeuge (den default nutzen)

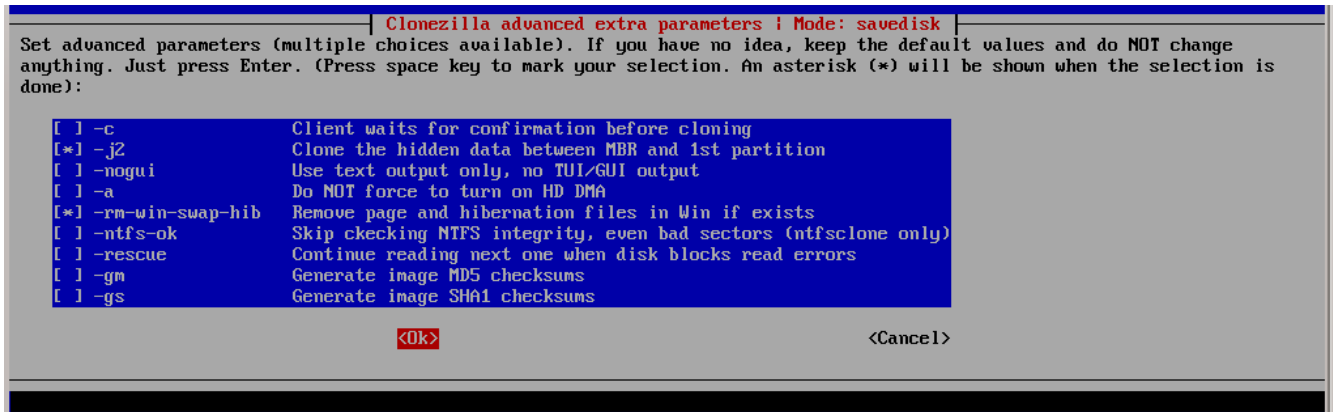


Abbildung 122: Durchmishtes: Hier -c abwählen um für die automation zusätzliche Rückfragen zu vermeiden.

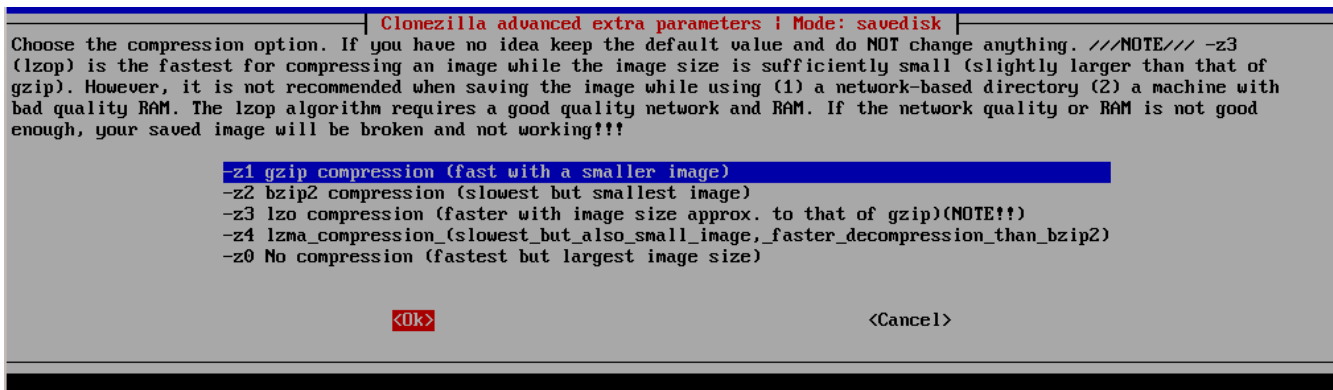


Abbildung 123: Kompressionsmethode vor opsi 4.0.5 also bei bootimages ⇐ 20130207 (opsi-clonezilla_2.01-3) hier -z1 wählen. Ab opsi 4.0.5 ist dies nicht mehr nötig.

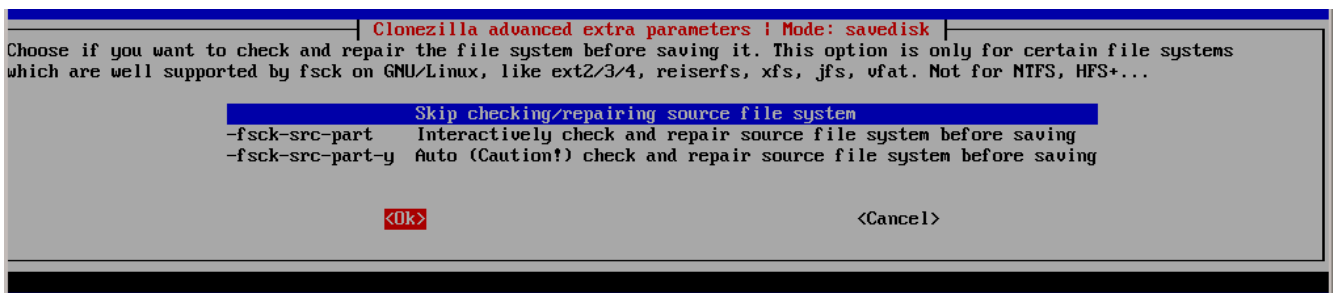
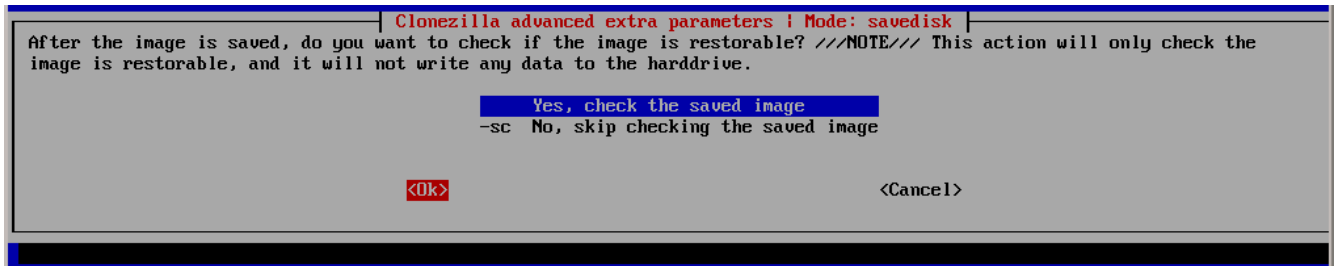
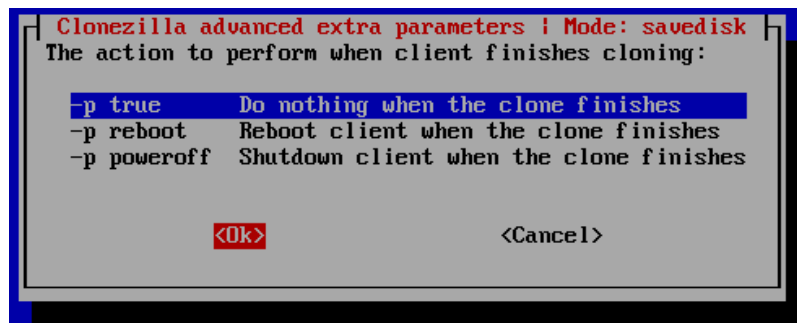


Abbildung 124: Check filesystem (den default *skip* nutzen)

Abbildung 125: Check the saved image (den default *yes* nutzen)Abbildung 126: Action after cloning (den default *-p true* nutzen, der reboot wird vom opsi bootimage ausgelöst.)

Interaktives savedisk

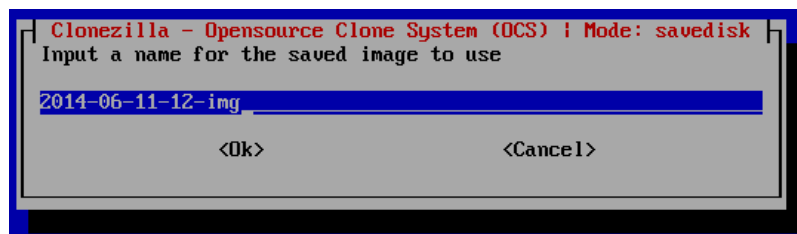


Abbildung 127: Name unter dem das image der disk abgespeichert werden soll.

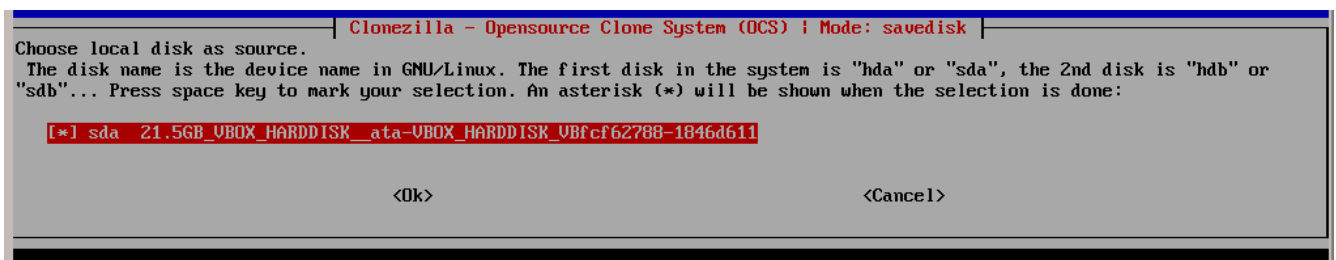


Abbildung 128: Wahl der disk von welcher das image erzeugt werden soll

```

*****
PS. Next time you can run this command directly:
/opt/drbl/sbin/ocs-sr -q2 -j2 -rm-win-swap-hib -z1 -i 2000 -p true savedisk 2014-06-11-12-img sda
This command is also saved as this file name for later use if necessary: /tmp/ocs-2014-06-11-12-img-2014-06-11-21
Press "Enter" to continue...

```

Abbildung 129: Das resultierende Kommando. Dieses kann auch im Produktproperty *runcommand* verwendet werden

```

Reading the partition table for /dev/sda...RETOAL=0
*****
The first partition of disk /dev/sda starts at 2048.
Saving the hidden data between MBR (1st sector, i.e. 512 bytes) and 1st partition, which might
by:
dd if=/dev/sda of=/home/partimag/partimg/sda-hidden-data-after-mbr skip=1 bs=512 count=2047
2047+0 records in
2047+0 records out
1048064 bytes (1,0 MB) copied, 0,0257349 s, 40,7 MB/s
*****
done!
Saving the MBR data for sda...
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0,00170123 s, 301 kB/s
*****
Starting saving /dev/sda1 as /home/partimag/partimg/sda1.XXX...
/dev/sda1 filesystem: ntfs.
*****
Checking the disk space...
*****
Use partclone with gzip to save the image.
Image file will be split with size limit 2000 MB.
*****
If this action fails or hangs, check:
* Is the disk full ?
*****
Partclone v0.2.8 http://partclone.org
Starting to clone device (/dev/sda1) to image (-)
Reading Super Block
Calculating bitmap...
Elapsed: 00:00:01, Remaining: 00:00:00, Completed:100.00%,
Total Time: 00:00:01, 100.00% completed!
File system: NTFS
Device size: 21.5 GB
Space in use: 13.8 GB
Free Space: 7.7 GB
Block size: 4096 Byte
Used block : 3371140
Elapsed: 00:00:35, Remaining: 00:08:55, Completed: 6.14%, Rate: 1.45GB/min,

```

Abbildung 130: Fortschrittsanzeige

Interaktives savepart

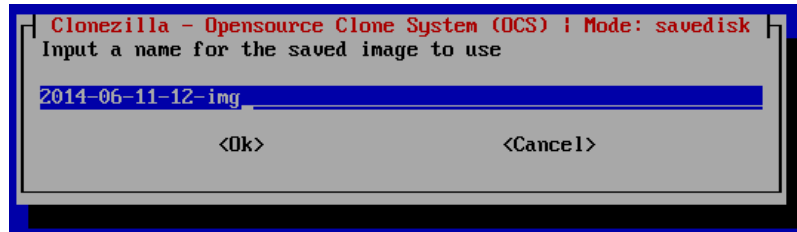


Abbildung 131: Name unter dem das image der partition abgespeichert werden soll.

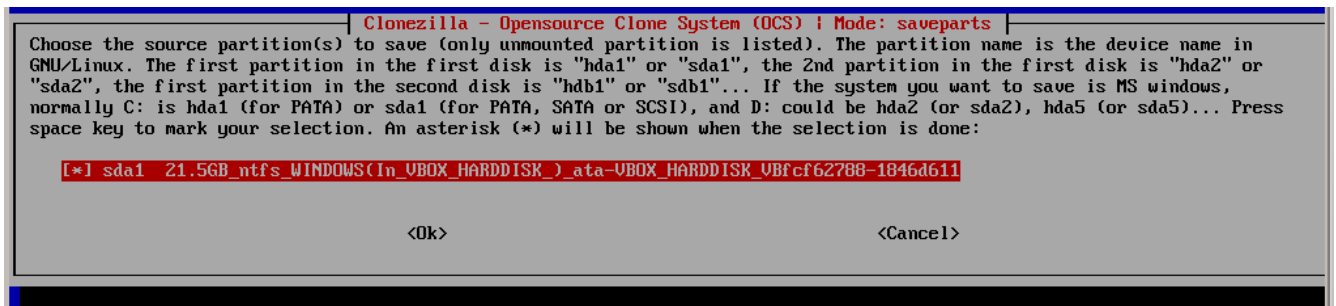


Abbildung 132: Wahl der partition von welcher das image erzeugt werden soll

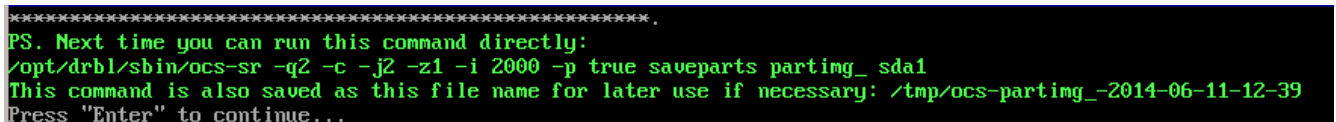


Abbildung 133: Das resultierende Kommando. Dieses kann auch im Produktproperty *runcommand* verwendet werden

```

Reading the partition table for /dev/sda...RETOVAL=0
*****.
The first partition of disk /dev/sda starts at 2048.
Saving the hidden data between MBR (1st sector, i.e. 512 bytes) and 1st partition, which mig
by:
dd if=/dev/sda of=/home/partimag/partimg/sda-hidden-data-after-mbr skip=1 bs=512 count=2047
2047+0 records in
2047+0 records out
1048064 bytes (1,0 MB) copied, 0,0257349 s, 40,7 MB/s
*****.
done!
Saving the MBR data for sda...
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0,00170123 s, 301 kB/s
*****.
Starting saving /dev/sda1 as /home/partimag/partimg/sda1.XXX...
/dev/sda1 filesystem: ntfs.
*****.
Checking the disk space...
*****.
Use partclone with gzip to save the image.
Image file will be split with size limit 2000 MB.
*****.
If this action fails or hangs, check:
* Is the disk full ?
*****.
Partclone v0.2.8 http://partclone.org
Starting to clone device (/dev/sda1) to image (-)
Reading Super Block
Calculating bitmap...
Elapsed: 00:00:01, Remaining: 00:00:00, Completed:100.00%,
Total Time: 00:00:01, 100.00%% completed!
File system: NTFS
Device size: 21.5 GB
Space in use: 13.8 GB
Free Space: 7.7 GB
Block size: 4096 Byte
Used block : 3371140
Elapsed: 00:00:35, Remaining: 00:08:55, Completed: 6.14%, Rate: 1.45GB/min,

```

Abbildung 134: Fortschrittsanzeige

Interaktives restoredisk

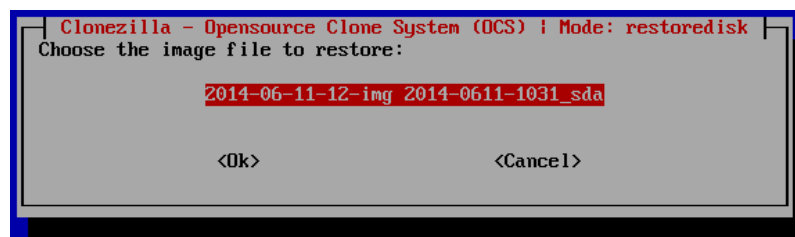


Abbildung 135: Wahl des diskimage welches restored werden soll

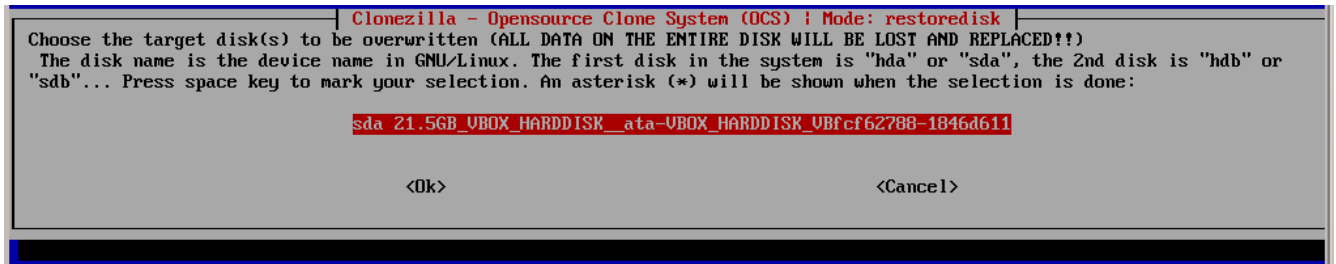


Abbildung 136: Wahl der disk auf die das image restored werden soll

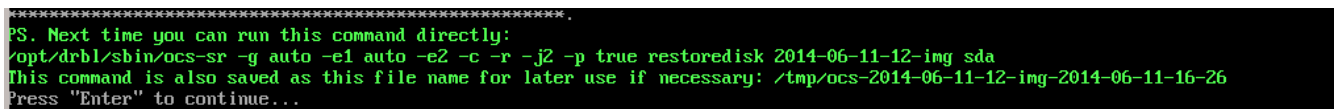
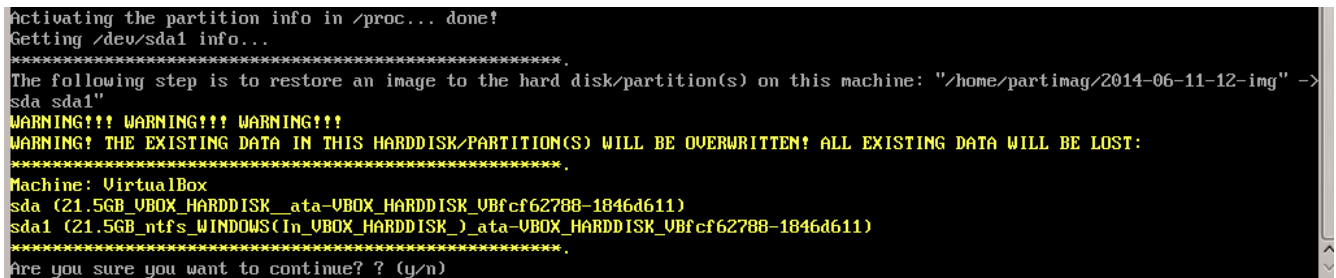
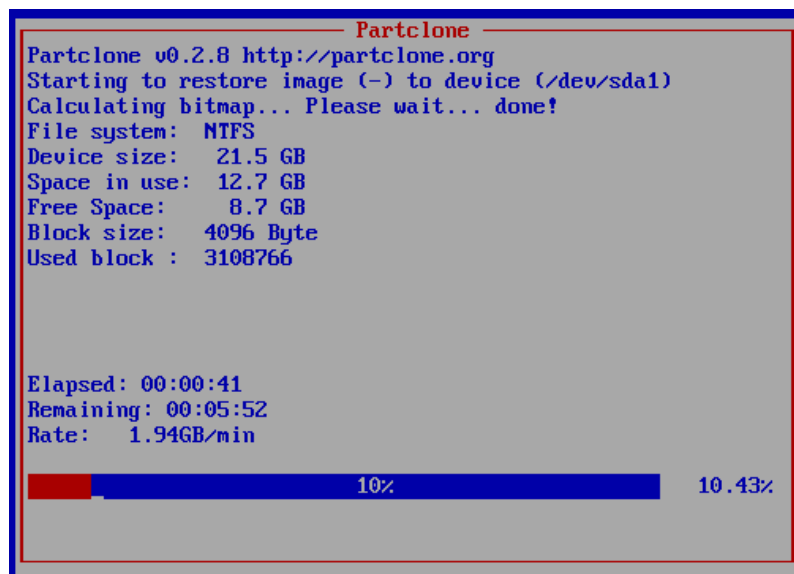
Abbildung 137: Das resultierende Kommando. Dieses kann auch im Produktproperty *runcommand* verwendet werdenAbbildung 138: Rückfrage vor dem Beginn des Überschreibens der disk. Kann durch entfernen der Option *-c* aus dem Kommando unterdrückt werden

Abbildung 139: Fortschrittsanzeige

Interaktives restorepart

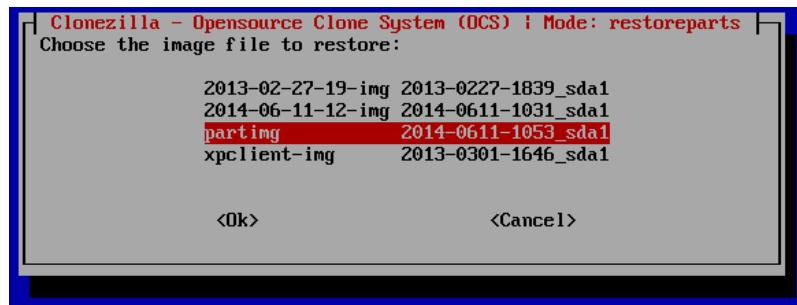


Abbildung 140: Wahl des partimage welches restored werden soll

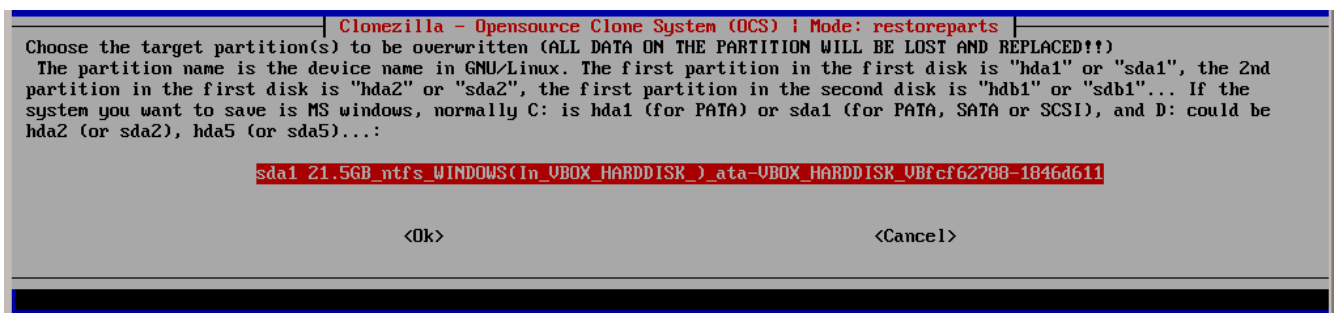


Abbildung 141: Wahl der partition auf die das image restored werden soll

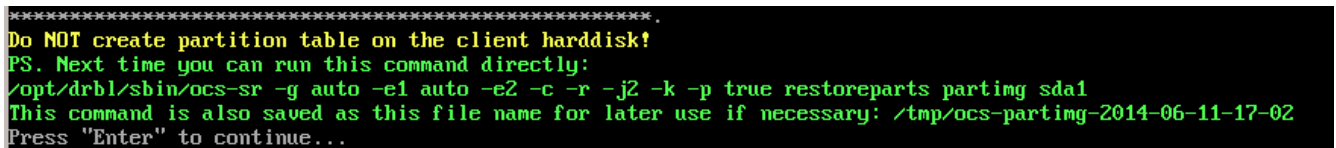


Abbildung 142: Das resultierende Kommando. Dieses kann auch im Produktproperty *runcommand* verwendet werden

```

Activating the partition info in /proc... done!
Getting /dev/sda1 info...
*****
The following step is to restore an image to the hard disk/partition(s) on this machine: "/home/partimag/partimg" -> "sda sda1
WARNING!!! WARNING!!! WARNING!!!
WARNING! THE EXISTING DATA IN THIS HARDDISK/PARTITION(S) WILL BE OVERWRITTEN! ALL EXISTING DATA WILL BE LOST:
*****
Machine: VirtualBox
sda1 (21.5GB_ntfs_WINDOWS(In_VBOX_HARDDISK_)_ata-VBOX_HARDDISK_VBfcf62788-1846d611)
*****
Are you sure you want to continue? ? (y/n) y
OK, let's do it!!
This program is not started by clonezilla server.
The following step is to restore an image to the hard disk/partition(s) on this machine: "/home/partimag/partimg" -> "sda (sda
"
WARNING!!! WARNING!!! WARNING!!!
WARNING! THE EXISTING DATA IN THIS HARDDISK/PARTITION(S) WILL BE OVERWRITTEN! ALL EXISTING DATA WILL BE LOST:
*****
Machine: VirtualBox
sda1 (21.5GB_ntfs_WINDOWS(In_VBOX_HARDDISK_)_ata-VBOX_HARDDISK_VBfcf62788-1846d611)
*****
Let me ask you again. Are you sure you want to continue? ?
[y/n]

```

Abbildung 143: Rückfrage vor dem Beginn des Überschreibens der disk. Kann durch entfernen der Option `-c` aus dem Kommando unterdrückt werden.

```

----- Partclone -----
Partclone v0.2.8 http://partclone.org
Starting to restore image (-) to device (/dev/sda1)
Calculating bitmap... Please wait... done!
File system: NTFS
Device size: 21.5 GB
Space in use: 13.8 GB
Free Space: 7.7 GB
Block size: 4096 Byte
Used block : 3371140

Elapsed: 00:00:16
Remaining: 00:05:56
Rate: 2.22GB/min

4% 4.29%

```

Abbildung 144: Fortschrittsanzeige

9.12.5 Nichtinteraktive Abläufe

Durch Angabe des entsprechenden Kommandos im Property `runcommand` wird opsi-clonezilla in den nichtinteraktiven Modus versetzt.

- Stellen Sie das Property `imageshare` auf einen share, welcher vom user `pcpatch` mit dem dem opsi-server bekannten Passwort gemountet werden kann. Das Format für den share ist dabei `//server/share` (Beachten Sie die Verwendung von slashes statt backslashes).
- Stellen Sie das Property `runcommand` auf ein entsprechendes nicht-interaktives Kommando. Dabei empfohlene Schalter:
 - Immer: `--batch`
 - Beim Restore: `--skip-check-restorable-r`

- Weglassen: `-c`

Beispiele aus den bisher gezeigten Vorgängen (ohne `-c` und mit `--batch`). Ab opsi 4.0.5 können Sie den Parameter `-z1` weglassen. Dies beschleunigt die Kompression bei mehreren Prozessor Kernen:

- `/opt/drbl/sbin/ocs-sr --batch -q2 -j2 -rm-win-swap-hib -z1 -i 2000 -p true savedisk 2014-06-11-12-img sda`
- `/opt/drbl/sbin/ocs-sr --batch -q2 -j2 -rm-win-swap-hib -z1 -i 2000 -p true saveparts partimg sda1`
- `/opt/drbl/sbin/ocs-sr --batch -g auto -e1 auto -e2 -r -j2 -p true restoredisk 2014-06-11-12-img sda`
- `/opt/drbl/sbin/ocs-sr --batch -g auto -e1 auto -e2 -r -j2 -k -p true restoreparts partimg sda1`

Wenn man nun bei diesen Beispielen die imagenamen `2014-06-11-12-img` bzw. `partimg` durch den string *imagefile* ersetzt, so wird dieser String jeweils durch den Wert des Properties `imagefile` ersetzt.

Wenn man nun bei diesen Beispielen die Devicenamen `sda` bzw. `sda1` durch den string *diskdevice* bzw. *partdevice* ersetzt, so wird dieser String jeweils durch einen für diese Maschine korrekten Wert gemäß der der Properties `disk_number` bzw. `part_number` ersetzt.

Beispiele für `disk_number=1` und `part_number=1`:

`sda / sda1`
`cciss/c0d0 / cciss/c0d0p1`

Daraus ergeben sich dann folgende Beispiele:

- `ocs-sr -g auto -e1 auto -e2 --skip-check-restorable-r --batch -r -j2 -p true restoredisk imagefile diskdevice`
- `ocs-sr -q2 --batch -j2 -rm-win-swap-hib -i 2000 -p true savedisk imagefile diskdevice`
- `ocs-sr -q2 -c -j2 -z1 -i 2000 -sc -p true saveparts imagefile partdevice`

9.12.6 opsi-clonezilla properties

- `askbeforeinst`
 - description: Should there be a confirmation dialog before start installing ? / Faut-il y avoir une confirmation avant de démarrer l'installation ?
 - default: False
- `mount_image_share`
 - description: Should there be a confirmation dialog before start installing ? / Faut-il y avoir une confirmation avant de démarrer l'installation ?
 - default: True
- `imageshare`

- editable: True
 - description: normally **auto** or **empty**: Defaults to the `opsi_images` share of the depot server; if not **auto** or **empty**: smb/cifs share in the format `//server/share`
 - values: ["", "//opsiserver/opsi_images", "auto"]
 - default: ["auto"]
 - **runcommand**
 - editable: True
 - description: Clonezilla command to be executed
 - values: ["", "ocs-live", "ocs-sr -g auto -e1 auto -e2 --skip-check-restorable-r --batch -r -j2 -p true restoredisk imagefile diskdevice", "ocs-sr -q2 --skip-check-restorable-s --batch -j2 -rm-win-swap-hib -i 2000 -p true savedisk imagefile diskdevice", "ocs-sr -q2 -c -j2 -z1 -i 2000 -sc -p true saveparts imagefile partdevice"]
 - default: ["ocs-live"]
 - **disk_number**
 - editable: True
 - description: Number (first=1) of the disk ; if string *diskdevice* in the runcommand it will be replaced by valid device path (eg sda)
 - values: ["1", "2"]
 - default: ["1"]
 - **part_number**
 - editable: True
 - description: Number (first=1) of the partition of *disk_number* ; if string *partdevice* in the runcommand it will be replaced by valid device path (eg sda1)
 - values: ["1", "2", "3", "4", "5"]
 - default: ["1"]
 - **imagefile**
 - editable: True
 - description: name of the imagefile ; will replace the string *imagefile* in the runcommand
 - values: ["myimagefile"]
 - default: ["myimagefile"]
 - **drbl_ocs_conf**
 - editable: True
 - description: Directory for post run scripts (Entries in `/etc/drbl/drbl-ocs.conf`)
 - values: ["", "OCS_POSTRUN_DIR=\"/home/partimag/postrun\"", "OCS_PRERUN_DIR=\"/home/partimag/prerun\""]
 - **rebootflag**
 - editable: False
 - description: Should the Client reboot after running the script
-

- values: ["keepalive", "reboot", "shutdown"]
- default: ["reboot"]
- **setup_after_install**
 - multivalue: True
 - editable: True
 - description: Which opsi product(s) should we switch to setup after clonezilla work is finished ?
 - values: [""]
 - default: [""]
- **architecture**
 - editable: False
 - description: Architektur Auswahl, beeinflusst die Auswahl des bootimages und die Installationsarchitektur.
 - values: ["32bit", "64bit"]
 - default: ["32bit"]

9.12.7 opsi-clonezilla known bugs

Keine

9.12.8 Clonezilla Kommando Referenz

Sichern oder Wiederherstellen von Images

[Clonezilla live doc](#)

```
Clonezilla ocs-sr options
```

```
/usr/sbin/ocs-sr:
Usage:
To save or restore image
ocs-sr [OPTION] {savedisk|saveparts|restoredisk|restoreparts} IMAGE_NAME DEVICE
```

Options for saving:

-enc, --enc-ocs-img

To encrypt the image with passphrase.

-fsck-src-part, --fsck-src-part

Run fsck interactively on the source file system before saving it.

-fsck-src-part-y, --fsck-src-part-y

Run fsck automatically on the source file system before saving it. This option will always attempt to fix any detected filesystem corruption automatically. //NOTE// Use this option in caution.

-gm, --gen-md5sum

Generate the MD5 checksum for the image. Later you can use -cm|--check-md5sum option to check the image when restoring the image. Note! It might take a lot of time to generate if the image size is large.

-gs, --gen-sha1sum

Generate the SHA1 checksum for the image. Later you can use -cs|--check-sha1sum option to check the image when restoring the image. Note! It might take a lot of time to generate if the image size is large.

-gmf, --gen-chksum-for-files-in-dev

Generate the checksum for files in the source device. Later you can use `-cmf|--chk-chksum-for-files-in-dev` to check the files in the destination device after they are restored. Note! It might take a lot of time to inspect the checksum if there are many files in the destination device.

-i, --image-size SIZE

Set the size in MB to split the partition image file into multiple volumes files. For the FAT32 image repository, the SIZE should not be larger than 4096.

-j2, --clone-hidden-data

Use dd to clone the image of the data between MBR (1st sector, i.e. 512 bytes) and 1st partition, which might be useful for some recovery tool.

-ntfs-ok, --ntfs-ok

Assume the NTFS integrity is OK, do NOT check again (for ntfsclone only)

-rm-win-swap-hib, --rm-win-swap-hib

Try to remove the MS windows swap file in the source partition.

-q, --use-ntfsclone

If the partition to be saved is NTFS, use program ntfsclone instead of partimage (i.e. Priority: ntfsclone > partimage > dd)

-q1, --force-to-use-dd

Force to use dd to save partition(s) (inefficient method, very slow, but works for all the file system).

-q2, --use-partclone

Use partclone to save partition(s) (i.e. partclone > partimage > dd).

-rescue, --rescue

Turn on rescue mode, i.e. try to skip bad sectors.

-sc, -scs, --skip-check-restorable, --skip-check-restorable-s

By default Clonezilla will check the image if restorable after it is created. This option allows you to skip that.

-z0, --no-compress

Don't compress when saving: very fast but very big image file (NOT compatible with multicast restoring!!!)

-z1, --gzip-compress

Compress using gzip when saving: fast and small image file (default)

-z1p, --smp-gzip-compress

Compress using parallel gzip program (pigz) when saving: fast and small image file, good for multi-core or multi-CPU machine

-z2, --bz2-compress

Compress using bzip2 when saving: slow but smallest image file

-z2p, --smp-bzip2-compress

Compress using parallel bzip2 program (lbzip2) when saving: faster and smallest image file, good for multi-core or multi-CPU machine

-z3, --lzo-compress

Compress using lzop when saving: similar to the size by gzip, but faster than gzip.

-z4, --lzma-compress

Compress using lzma when saving: slow but smallest image file, faster decompression than bzip2.

-z5, --xz-compress

Compress using xz when saving: slow but smallest image file, faster decompression than bzip2.

-z5p, --smp-xz-compress

Compress using parallel xz when saving: slow but smallest image file, faster decompression than bzip2.

-z6, --lzip-compress

Compress using lzip when saving: slow but smallest image file, faster decompression than bzip2.

-z6p, --smp-lzip-compress

Compress using parallel lzip when saving: slow but smallest image file, faster decompression than bzip2.

-z7, --lrzip-compress

Compress using lrzip when saving.

-i, --image-size SIZE

Set the split image file volume size SIZE (MB). When ocs-sr is run with -x, the default SIZE is set as 4096, if without -x, we will not split it. Some words are reserved for IMAGE_NAME, "ask_user" is used to let user to input a name when saving an image. "autoname" is used to automatically generate the image name based on network card MAC address and time. "autohostname" is used to automatically generate the image name based on hostname. "autoproductname" is used to automatically generate the image name based on hardware product model gotten from dmidecode. A word is reserved for DEVICE, "ask_user" could be used to let user to select the source device when saving an image.

Options for restoring:

-f, --from-part-in-img PARTITION

Restore the partition from image. This is especially for "restoreparts" to restore the image of partition (only works for one) to different partition, e.g. sda1 of image to sdb6.

-g, --grub-install GRUB_PARTITION

Install grub in the MBR of the disk containing partition GRUB_PARTITION with root grub directory in the same GRUB_PARTITION when restoration finishes, GRUB_PARTITION can be one of "/dev/hda1", "/dev/hda2"... or "auto" ("auto" will let clonezilla detect the grub root partition automatically). If "auto" is assigned, it will work if grub partition and root partition are not in the same partition.

-r, --resize-partition

Resize the partition when restoration finishes, this will resize the file system size to fit the partition size. It is normally used when when a small partition image is restored to a larger partition.

-k, --no-fdisk, --no-create-partition

Do NOT create partition in target harddisk. If this option is set, you must make sure there is an existing partition table in the current restored harddisk. Default is to create the partition table.

-icrc, --icrc

Skip Partclone CRC checking.

-irhr, --irhr

Skip removing the Linux udev hardware records on the restored GNU/Linux.

-irvd, --irvd

Skip removing the NTFS volume dirty flag after the file system is restored.

-ius, --ius

Skip updating syslinux-related files on the restored GNU/Linux.

-icds, --ignore-chk-dsk-size-pt

Skip checking destination disk size before creating the partition table on it. By default it will be checked and if the size is smaller than the source disk, quit.

iefi, --ignore-update-efi-nvram

Skip updating boot entries in EFI NVRAM after restoring.

-k1,

Create partition table in the target disk proportionally.

-k2,

Enter command line prompt to create partition table manually before restoring image.

-scr, --skip-check-restorable-r

By default Clonezilla will check the image if restorable before restoring. This option allows you to skip that.

-t, --no-restore-mbr

Do NOT restore the MBR (Master Boot Record) when restoring image. If this option is set, you must make sure there is an existing MBR in the current restored harddisk. Default is Yes

-u, --select-img-in-client

Input the image name in clients

-e, --load-geometry

Force to use the saved CHS (cylinders, heads, sectors) when using sfdisk

-e1, --change-geometry NTFS-BOOT-PARTITION

Force to change the CHS (cylinders, heads, sectors) value of NTFS boot partition after image is restored. NTFS-BOOT-PARTITION can be one of "/dev/hda1", "/dev/hda2"... or "auto" ("auto" will let clonezilla detect the NTFS boot partition automatically)

-e2, --load-geometry-from-edd

Force to use the CHS (cylinders, heads, sectors) from EDD (Enhanced Disk Device) when creating partition table by sfdisk

-y, -y0, --always-restore, --always-restore-default-local

Let Clonezilla server as restore server, i.e. client will always has restore mode to choose (However default mode in PXE menu is local boot)

-y1, --always-restore-default-clone

Let Clonezilla server as restore server, i.e. client will always has restore mode to choose (The default mode in PXE menu is clone, so if client boots, it will enter clone always, i.e. clone forever)

-j, --create-part-by-sfdisk

Use sfdisk to create partition table instead of using dd to dump the partition table from saved image (This is default)

-j0, --create-part-by-dd

Use dd to dump the partition table from saved image instead of sfdisk. ///Note/// This does NOT work when logical drives exist.

-j1, --dump-mbr-in-the-end

Use dd to dump the MBR (total 512 bytes, i.e. 446 bytes (executable code area) + 64 bytes (table of primary partitions) + 2 bytes (MBR signature; # 0xAA55) = 512 bytes) after disk image was restored. This is an insurance for some hard drive has different numbers of cylinder, head and sector between image was saved and restored.

-j2, --clone-hidden-data

Use dd to clone the image of the data between MBR (1st sector, i.e. 512 bytes) and 1st partition, which might be useful for some recovery tool.

-hn0 PREFIX

Change the hostname of M\$ Windows based on the combination of hostname prefix and IP address, i.e. PREFIX-IP

-hn1 PREFIX

Change the hostname of M\$ Windows based on the combination of hostname prefix and NIC MAC address, i.e. PREFIX-MAC

--max-time-to-wait TIME

When not enough clients have connected (but at least one), start anyways when TIME seconds since first client connection have passed. This option is used with --clients-to-wait

-cm, --check-md5sum

Check the MD5 checksum for the image. To use this option, you must enable -gm|--gen-md5sum option when the image is saved. Note! It might take a lot of time to check if the image size is large.

-cs, --check-sha1sum

Check the SHA1 checksum for the image. To use this option, you must enable `-gs|--gen-sha1sum` option when the image is saved. Note! It might take a lot of time to check if the image size is large.

-cmf, --chk-chksum-for-files-in-dev

Check the checksum for the files in the device. To use this option, you must enable `-gmf|--gen-chksum-for-files-in-dev` when the image is saved. Note! (1) The file system must be supported by Linux kernel so that it can be mounted as read-only to check the files. (2) It might take a lot of time to check if there are many files in the source device.

-srel, --save-restore-error-log

Save the error log file in the image dir. By default the log file won't be saved when error occurs.

--mcast-port NO

Assign the udp port number for multicast restore. This is used by clonezilla server. Normally it's not necessary to manually assign this option. Some words are reserved for `IMAGE_NAME`, "ask_user" is used to let user to input a name when saving an image. "autoproduckname" is used to automatically get the image name based on hardware product model from `dmidecode`. A word is reserved for `DEVICE`, "ask_user" could be used to let user to select the source device when saving an image.

General options:

l, --language INDEX

Set the language to be shown by index number: [0|en_US.UTF-8]: English, [1|zh_TW.BIG5]: Traditional Chinese (Big5) - Taiwan, [2|zh_TW.UTF-8]: Traditional Chinese (UTF-8, Unicode) - Taiwan [a|ask]: Prompt to ask the language index

-b, -batch, --batch

(DANGEROUS!) Run program in batch mode, i.e. without any prompt or wait for pressing enter key. //NOTE// You have to use `-batch` instead of `-b` when you want to use it in the boot parameters. Otherwise the program init on system will honor `-b`, too.

-c, --confirm

Wait for confirmation before saving or restoring

-d, --debug-mode

Enter command mode to debug before saving/restoring

--debug=LEVEL

Output the partimage debug log in directory `/var/log/` with debug LEVEL (0,1,2... default=0)

-m, --module MODULE

Force to load kernel module MODULE, this is useful when some SCSI device is not detected. NOTE! Use only one module, more than one may cause parsing problem.

-o0, --run-prerun-dir

Run the script in the directory `/usr/share/drbl/postrun/ocs/` before clone is started. The command will be run before MBR is created or saved.

-o1, -o, --run-postrun-dir

Run the script in the directory `/usr/share/drbl/postrun/ocs/` when clone is finished. The command will be run before that assigned in `-p` or `--postaction`.

-w, --wait-time TIME

Wait for TIME secs before saving/restoring

-nogui, --nogui

Do not show GUI (TUI) of Partclone or Partimage, use text only

-a, --no-force-dma-on

Do not force to turn on HD DMA

-mp, --mount-point MOUNT_POINT

Use NFS to mount MOUNT_POINT as directory ocsroot (ocsroot is assigned in drbl.conf)

-or, --ocsroot DIR

Specify DIR (absolute path) as directory ocsroot (i.e. overwrite the ocsroot assigned in drbl.conf)

-p, --postaction [choose|poweroff|reboot|command|CMD]

When save/restoration finishes, choose action in the client, poweroff, reboot (default), in command prompt or run CMD

-ns, --ntfs-progress-in-image-dir

Save the ntfsclone progress tmp file in the image dir so that if cloning is in DRBL client, the progress can be check in the server (Default in to be put in local /tmp/, which is local tmpfs).

-um, --user-mode [beginner|expert]

Specify the mode to use. If not specified, default mode is for a beginner.

-v, --verbose

Prints verbose information

-d0, --dialog

Use dialog

-d1, --Xdialog

Use Xdialog

-d2, --whiptail

Use whiptail

-d3, --gdialog

Use gdialog

-d4, --kdialog

Use kdialog

-x, --interactive

Interactive mode to save or restore.

Example:

- To save or restore image in client (Only that DRBL client will join, and its local partitions is NOT mounted). NOTE!!! You should run the command in DRBL client or you have to make sure the target device is NOT busy!. To save all the data in local first IDE harddrive *hda* as image *IMAGE1*, use ntfsclone instead of partimage, and lzop compression (NOTE!!! You should run the command in DRBL client or make sure hda is NOT busy/mounted!): `ocs-sr --use-ntfsclone -z3 savedisk IMAGE1 hda`
- To save the data in first and second partitions in local first IDE harddrive *hda* as image *IMAGE2*, use ntfsclone instead of partimage, and lzop compression (NOTE!!! You should run the command in DRBL client, or make sure hda is NOT busy/mounted!): `ocs-sr --use-ntfsclone -z3 saveparts IMAGE2 "hda1 hda2"`
- To restore image *IMAGE1* to local *hda*. grub-install will be run after cloning (image *IMAGE1* is already in DRBL server. NOTE!!! You should run the command in DRBL client or make sure hda is NOT busy/mounted!): `ocs-sr -g auto restoredisk IMAGE1 hda`
- To restore image first and second partitions from *IMAGE2* to local *hda1* and *hda2*. grub-install will be run after cloning (image *IMAGE2* is already in DRBL server. NOTE!!! You should run the command in DRBL client or make sure hda is NOT busy/mounted!): `ocs-sr -g auto restoreparts IMAGE2 "hda1 hda2"`
- To save disk(s)/partition(s) as an image or restore an image to disk(s)/partition(s) interactively, use: `ocs-sr -x`

disk-to-disk Operation

DRBL management: Clone disk or partition on-the-fly

Clone disk or partition on-the-fly

The "ocs-onthefly" is used to do disk to disk or partition to partition copy on-the-fly. This command is different from drbl-ocs (or clonezilla). Clonezilla is used to do massively clone, so it will save the template machine as an image in clonezilla server. On the other hand, ocs-onthefly is used to 1 to 1 copy, so no image will be saved in the server. Just clone disk or partition directly.

There are 2 ways to run ocs-onthefly:

1. Clone locally: Boot the machine as DRBL client, then clone one disk to another disk. This is specially for when you just want to clone disk, and you only have one machine.
2. Clone via network: Boot the source and target machine as DRBL clients, then clone disk from one machine to another machine. This is specially for you have 2 machines, and you want to clone them without dismantling machine.

Usage:

ocs-onthefly [OPTION]

Option:

-e, --resize-partition

resize the target disk in target machine (To solve the small partition image restored to larger partition problem.)

-f, --source DEV

specify the source device as DEV (hda, hda1...)

-g, --grub-install GRUB_PARTITION

install grub in hda with root grub directory in GRUB_PARTITION when restoration finishes, GRUB_PARTITION can be one of "/dev/hda1", "/dev/hda2"... or "auto" ("auto" will clonezilla detects the grub root partition automatically)

-i, --filter PROGRAM

use the PROGRAM (gzip/lzop/bzip2/cat) before sending partition data to netcat (only in network clone mode). The default action is gzip. Use "cat" if you do not want to compress (Good for fast internode network).

-n, --no-sfdisk

skip partition table creation

-m, --no-mbr-clone

do NOT clone MBR

-o, --load-geometry

force to use the saved CHS (cylinders, heads, sectors) when using sfdisk in restoring.

-p, --port PORT

specify the netcat port (Only in network clone mode)

-r, --server

specify the running machine is in network clone server.

-s, --source-IP

IP specify the source IP address (used in target client machine).

-t, --target DEV

specify the target device as DEV (hda, hda1...)

-v, --verbose

prints verbose information

ocs-onthefly [OPTION]

Examples:

1. Clone locally: To clone the 1st harddisk (hda) to 2nd harddisk (hdb), you can boot this machine as DRBL client, then run:
`ocs-onthefly -f hda -t hdb`
2. Clone via network: To clone the 1st harddisk (hda) in machine A to the 1st harddisk (hda) in machine B. Then without dismantling machines, you can do it by:
 Boot machine A as DRBL client, and it's IP address is, say, 192.168.100.1, then run:
`ocs-onthefly -r -f hda`

Then it will prompt you the command to run in machine B, such as:

`ocs-thefly --source-IP 192.168.100.1 -t [TARGET_DEV]` (TARGET_DEV is like hda, hdb, hda1, hdb1...)

The "TARGET_DEV" is the target harddisk in machine B, in this case, it hda. Then, boot machine B as DRBL client, and run:

`ocs-onthefly --source-IP 192.168.100.1 -t hda`

9.13 opsi-server mit mehreren Depots (frei)

9.13.1 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 (configserver) gehalten.
- Alle Clients verbinden sich über den opsi-Webservice mit dem configserver und erhalten von dort ihre Konfigurationsinformationen.
- Die Softwaredepots liegen auf dezentralen *opsi-depotservern* und werden dem zentralen configserver als Netzwerkmounts zur Installation von Paketen zur Verfügung gestellt.
- Die Funktionalität zum Start von Bootimages mittels PXE wird ebenfalls auf dem dezentralen opsi-depotserver installiert. Diese wird aber zentral gesteuert.
- opsi-package-manager: Programm zur Unterstützung von mehreren Depotshares beim Installieren und Deinstallieren von opsi-Paketen.
- Transport der opsi-Pakete via webdav auf die opsi-depotserver 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 Produkt-Controllfiles.
- Ermöglichung der Selektion einzelner oder mehrerer Depotshares zur Auswahl der opsi-clients im opsi-configed.
- Unterbinden der gemeinsamen Bearbeitung von opsi-clients, die an Depotshares hängen und zueinander inkonsistent sind.

- Zuordnung der opsi-clients zu Depotshares über den opsi-configed, Umzug von Clients.
- Konfigurations- und Verbindungsdaten der einzelnen Depotshares über den opsi-configed editierbar machen.

Die folgenden Schemata geben einen Überblick über die Kommunikation zwischen den Komponenten bei einer Situation mit einem Standort und der Situation mit einem opsi-depotserver.

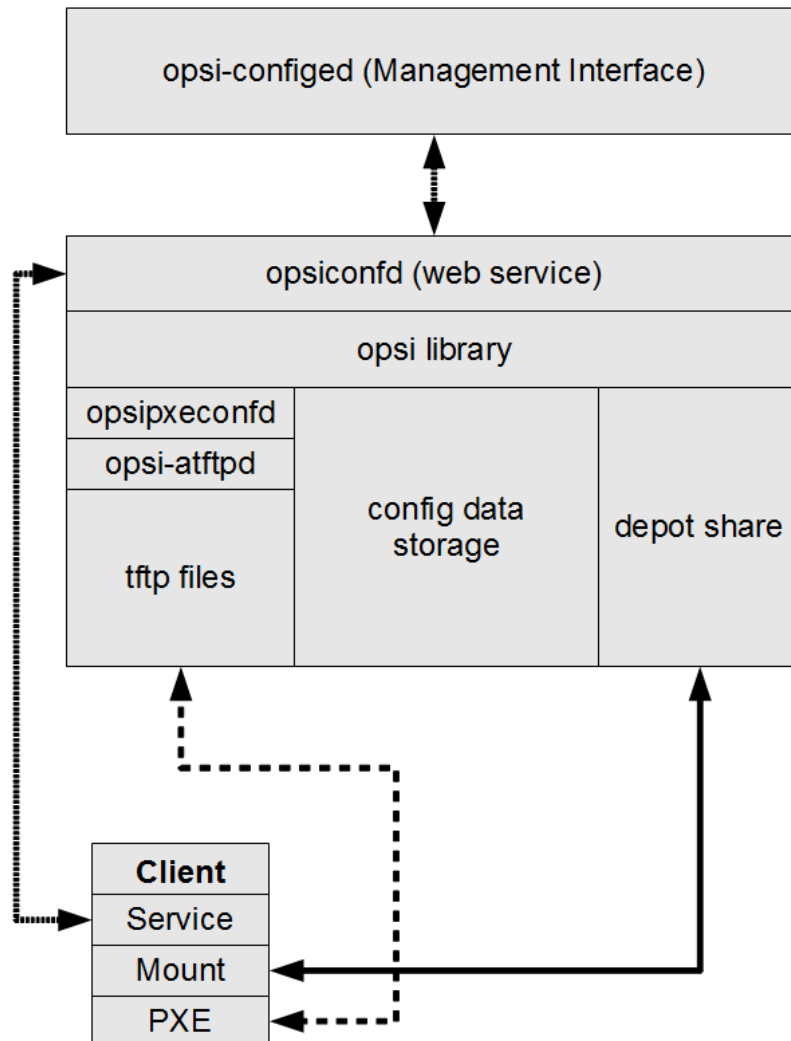


Abbildung 145: Schema: Kommunikation zwischen opsi-client und opsi-server (ein Standort)

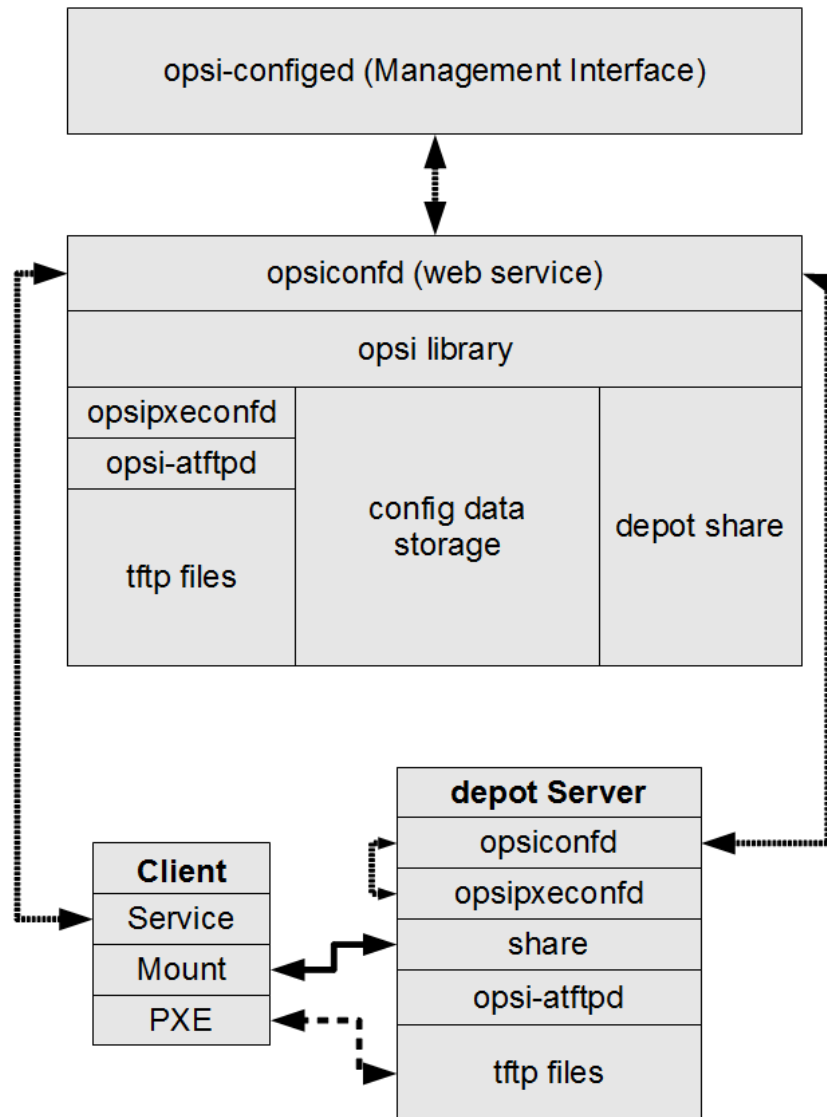


Abbildung 146: Schema: Kommunikation zwischen opsi-client und opsi-servern (mehrere Standorte)

9.13.2 Erstellung und Konfiguration eines Depot-Servers

Zur Erstellung eines externen *opsi-depotserver*s wird zunächst ein normaler *opsi-server* aufgesetzt. Dann wird auf diesem neuen *opsi-server* der Befehl `opsi-setup --register-depot` mit root Rechten ausgeführt, um ihn zum externen *opsi-depotserver* zu konfigurieren. Da hierbei nicht nur der *opsi-depotserver* konfiguriert wird, sondern dieser auch noch per Webservice dem zentralen *opsi-configserver* bekannt gemacht wird, müssen Username und Passwort eines Mitgliedes der Gruppe *opsiadmin* eingegeben werden.

Unter Univention Corporate Server findet die Registrierung der opsi-depotserver automatisch statt. Hierbei wird der erste Server mit einer opsi-Installation als opsi-configserver verwendet und alle weiteren in einer UCS-Domäne installierten Server werden dort als opsi-depotserver registriert.

Beispiel:

`svmdepotde.svm.local` wird als opsi-depotserver für den opsi-configserver `sepiella.svm.local` eingerichtet:

```
root@svmdepotde.svm.local:~# opsi-setup --register-depot
```

Nun erscheint die Maske zu dem opsi-configserver an dem sich dieser Server als opsi-depotserver anmelden soll. Diese Anmeldung muss mit einem User autorisiert werden, welcher auf dem opsi-configserver Mitglied in der Gruppe *opsiadmin* ist.

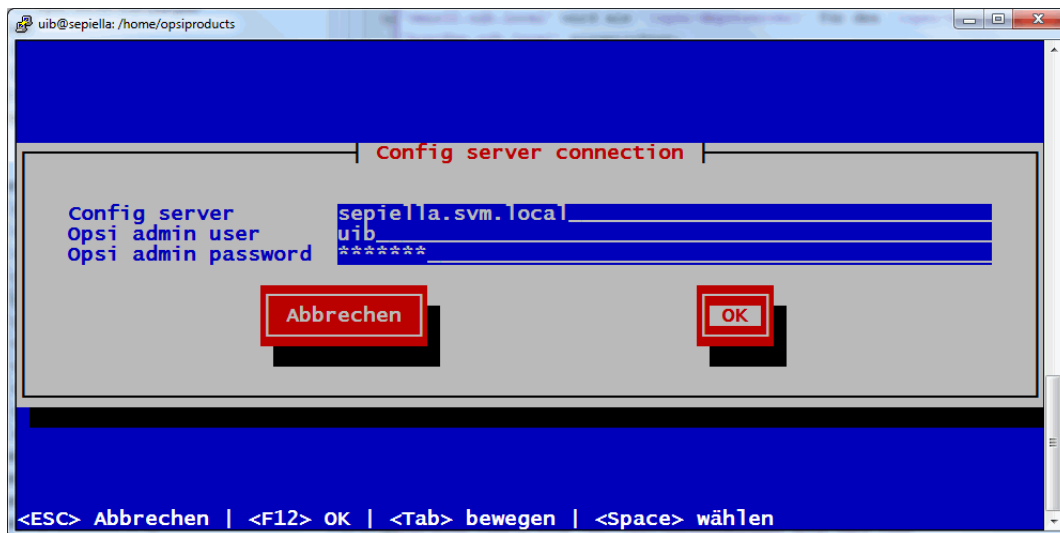


Abbildung 147: opsi-setup --register-depot : Maske Eingabe opsiadmin Account für *opsi-configserver*

Nun erscheint die Maske der Depotserver Settings. Im Normalfall müssen Sie hier nichts ändern. Insbesondere bleibt der neue opsi-depotserver in der Regel ein "Master-Depot", damit ihm anschließend *opsi-clients* zugeordnet werden können.

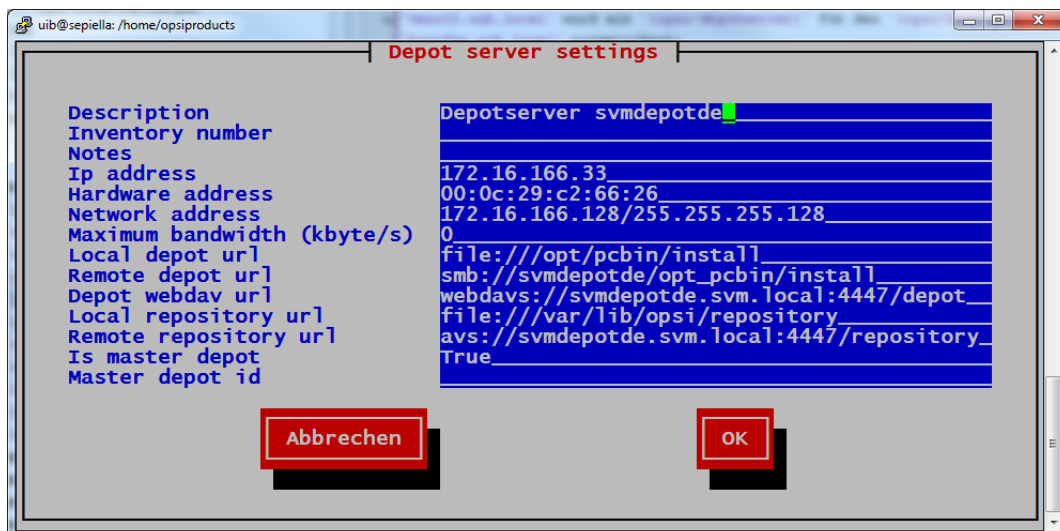


Abbildung 148: opsi-setup --register-depot : Maske Depot Settings

Nach dieser Eingabe der Daten erfolgt die eigentliche Konfiguration:

```
[5] [Apr 06 12:32:19] Getting current system config (opsi-setup|70)
[5] [Apr 06 12:32:19] System information: (opsi-setup|117)
[5] [Apr 06 12:32:19] distributor : Debian (opsi-setup|118)
```

```

[5] [Apr 06 12:32:19] distribution : Debian GNU/Linux 5.0.8 (lenny) (opsi-setup|119)
[5] [Apr 06 12:32:19] ip address   : 172.16.166.33 (opsi-setup|120)
[5] [Apr 06 12:32:19] netmask    : 255.255.255.0 (opsi-setup|121)
[5] [Apr 06 12:32:19] subnet     : 172.16.166.0 (opsi-setup|122)
[5] [Apr 06 12:32:19] broadcast  : 172.16.166.255 (opsi-setup|123)
[5] [Apr 06 12:32:19] fqdn       : svmdepotde.svm.local (opsi-setup|124)
[5] [Apr 06 12:32:19] hostname   : svmdepotde (opsi-setup|125)
[5] [Apr 06 12:32:19] domain     : svm.local (opsi-setup|126)
[5] [Apr 06 12:32:19] win domain  : OPSI (opsi-setup|127)
[5] [Apr 06 12:46:03] Creating depot 'svmdepotde.svm.local' (opsi-setup|2342)
[5] [Apr 06 12:46:03] Getting depot 'svmdepotde.svm.local' (opsi-setup|2345)
[5] [Apr 06 12:46:03] Testing connection to config server as user 'svmdepotde.svm.local' (opsi-setup|2354)
[5] [Apr 06 12:46:04] Successfully connected to config server as user 'svmdepotde.svm.local' (opsi-setup|2359)
[5] [Apr 06 12:46:04] Updating backend config '/etc/opsi/backends/jsonrpc.conf' (opsi-setup|2361)
[5] [Apr 06 12:46:04] Backend config '/etc/opsi/backends/jsonrpc.conf' updated (opsi-setup|2373)
[5] [Apr 06 12:46:04] Updating dispatch config '/etc/opsi/backendManager/dispatch.conf' (opsi-setup|2375)
[5] [Apr 06 12:46:04] Dispatch config '/etc/opsi/backendManager/dispatch.conf' updated (opsi-setup|2388)
[5] [Apr 06 12:46:04] Setting rights (opsi-setup|410)
[5] [Apr 06 12:46:06] Setting rights on directory '/tftpboot/linux' (opsi-setup|482)
[5] [Apr 06 12:46:06] Setting rights on directory '/home/opsiproducts' (opsi-setup|482)
[5] [Apr 06 12:46:06] Setting rights on directory '/var/log/opsi' (opsi-setup|482)
[5] [Apr 06 12:46:06] Setting rights on directory '/etc/opsi' (opsi-setup|482)
[5] [Apr 06 12:46:06] Setting rights on directory '/var/lib/opsi' (opsi-setup|482)
[5] [Apr 06 12:46:06] Setting rights on directory '/var/lib/opsi/depot' (opsi-setup|482)
[5] [Apr 06 12:46:27] Restarting services (opsi-setup|2392)
[5] [Apr 06 12:46:35] Configuring client user pcpatch (opsi-setup|347)
[5] [Apr 06 12:46:35] Creating RSA private key for user pcpatch in '/var/lib/opsi/.ssh/id_rsa' (opsi-setup|361)
[5] [Apr 06 12:46:35] Setting rights (opsi-setup|410)
[5] [Apr 06 12:46:38] Setting rights on directory '/var/lib/opsi/.ssh' (opsi-setup|482)

```

Es ist in der Regel nötig, die Konfigurationsdatei `opsi-product-updater.conf` auf dem neuen Depot zu überprüfen. Sofern der neue Depotserver sich lediglich opsi-Pakete vom zentralen Server holen soll, sollte nur eine solche Repo-Sektion aktiv bleiben:

```

[repository_master]
active = true
opsiDepotId = bonifax.uib.local
autoInstall = true
autoUpdate = true
autoSetup = false
; Inherit ProductProperty defaults from master repository
inheritProductProperties = false

```

Non-interaktive Registrierung eines opsi-depotserver

Es ist seit *opsi-depotserver* 4.0.7.2 möglich die Registrierung eines opsi-depotserver ohne Interaktion durchzuführen. Dabei müssen die Daten zur Verbindung an den opsi-configserver als JSON-Objekt mitsamt dem Parameter `--unattended` mitgegeben werden.

```
opsi-setup --register-depot --unattended '{"address": "config.server.address:4447/rpc", "username": "adminuserinopsi", \
"password": "pwoftheuser"}'
```

Der opsi-depotserver wird mit Standardwerten erstellt.

Es gibt die Möglichkeit benutzerdefinierte Attribute für den erstellten opsi-depotserver festzulegen. Dazu muss dem JSON-Objekt der Schlüssel `depot` und als Werte ein weiteres JSON-Objekt mit den gewünschten Werten mitgegeben werden.

Im folgenden Beispiel wird eine abweichende Beschreibung gesetzt:

```
opsi-setup --register-depot --unattended '{"address": "config.server.address:4447/rpc", "username": "adminuserinopsi", \
"password": "pwoftheuser", "depot": {"description": "Added with unattended registration."}}'
```

9.13.3 Paketmanagement auf mehreren Depots

siehe auch:

Abschnitt [5.3.2](#)

Abschnitt [5.3.3](#)

Zur Verwaltung der Pakete auf mehreren *opsi-depotserver* kennt der *opsi-package-manager* die Optionen `-d` bzw. `--depots` mit denen die *opsi-depotserver* 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/repository` des *opsi-depotserver* 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/repository` installiert.

Beispiel:

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

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

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

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
```

9.14 Dynamische Depotzuweisung (frei)

9.14.1 Einführung

Bei der Standard Multidepot Unterstützung in opsi, sind die Clients den jeweiligen Depots fest zu geordnet. Dies wird nun erweitert durch einen Mechanismus, mit dem ein Client erkennen kann, von welchem Depot er seine Software am schnellsten beziehen kann.

Eine Zuordnung gemäß IP-Nummern ist in vielen Bereichen die einfachste und passende Lösung. In anderen Netzwerktopologien reicht dies nicht aus, z.B. bei einem sternförmigen VPN-Netzwerk.

Notwendig ist also ein Mechanismus der Clientseitig dynamisch ermittelt, zu welchem Depot die beste Verbindung möglich ist. Die konkrete Implementation eines sinnvollen Algorithmus hierfür hängt wiederum sehr von der tatsächlichen Netzwerktopologie und den Wünschen des Kunden ab. Daher ist es sinnvoll diese konfigurierbar zu gestalten.

Ausgehend von der Überlegung, dass der Client sich nach den aktuellen Gegebenheiten des Netzwerks sein Depot sucht, ist sicherzustellen, dass die zur Auswahl stehenden Depots synchron, d.h. mit den selben Software-Paketen ausgestattet, sind. Da in der Praxis nicht alle Depots einer Multidepot-Umgebung immer synchron sein werden, wird die Liste der Depots, aus der sich ein Client das für ihn Geeignetste aussuchen kann, auf jene Depots beschränkt, die zum Masterdepots des Clients synchron sind. Das Masterdepot eines Clients ist das Depot, dem der Client zugewiesen ist. Damit bestimmt das Masterdepot, welche Software in welcher Version auf dem Client installiert werden kann.

Unser Konzept hierzu sieht wie folgt aus:

Auf dem opsi-configserver wird ein Client-Script hinterlegt, welches bei Bedarf auf den Client übertragen und dort interpretiert wird. Dieses Script entscheidet, welches der zur Verfügung stehenden Depots verwendet wird. Für dieses Clientscript ist definiert: die notwendige Schnittstelle mit dem das Script die Liste der zur Auswahl stehenden Server und aktuelle Client-Konfigurationen (IP-Adresse, Netzmaske, Gateway, ...) übernimmt und über die das Script dem Client das Ergebnis des Auswahlprozesses mitteilt, sowie Schnittstellen zum Logging sowie zur Anwenderinformation über den ablaufenden Prozess.

Die konkrete Implementation dieses Scriptes kann dann jederzeit an die konkrete Situation in der jeweiligen opsi-Umgebung angepasst werden.

Der aus diesem Konzept resultierende Ablauf eines Client-Connects sieht dann wie folgt aus:

1. Der Client meldet sich per Webservice beim opsi-configserver.
2. Der opsi-configserver übermittelt dem Client die Liste der zu installierenden Software.
3. Der opsi-configserver übermittelt dem Client das zentral abgelegte Script zur Auswahl des Depotserver sowie die Liste der möglichen Depots.
4. Der Client führt das Script aus und ermittelt damit das beste Depot.
5. Der Client verbindet sich mit dem ausgewählten Depotserver, um sich von dort die zu installierende Software zu holen.
6. Der Installationsstatus wird an den opsi-configserver zurückgemeldet.

9.14.2 Voraussetzungen

Diese Funktion setzt opsi in der Version ≥ 4.0 voraus.

Dieses Modul ist seit 1.3.2013 frei.

Der kofinanzierungs Prozess zur Finanzierung dieser opsi Erweiterung wurde im März 2013 abgeschlossen.

Weitere Details hierzu finden Sie in Abschnitt [9.1](#).

Diese Funktion benötigt mindestens folgende Paketstände:

Tabelle 25: Benötigte Pakete

opsi-Paket	Version
opsi-client-agent	$\geq 4.0-11$
opsi-configed	$\geq 4.0.1.5-1$
python-opsi	$\geq 4.0.0.18-1$

Anmerkung

Die Depotselektion wird über den opsi-client-agent realisiert. Im Umkehrschluss bedeutet dies, dass diese Funktion keine Anwendung bei Netbootprodukten findet.

9.14.3 Konfiguration

Das Script, welches der Client verwendet um die Depotauswahl durchzuführen, ist auf dem Server in der folgenden Datei abgelegt:

```
/etc/opsi/backendManager/extend.d/70_dynamic_depot.conf
```

Um die dynamische Depotauswahl für einen Client zu aktivieren, muss für diesen Client folgender Host-Parameter gesetzt werden:

```
clientconfig.depot.dynamic = true
```

Dies kann über den opsi-configed im Tab Host-Parameter geschehen.

Natürlich kann dies auch auf der Kommandozeile mit dem Befehl `opsi-admin` erledigt werden (`<client-id>` ist hierbei durch den FQDN, z.B. `client1.uib.local` des Clients zu ersetzen):

```
opsi-admin -d method configState_create \  
clientconfig.depot.dynamic <client-id> [True]
```

Kontrolliert werden kann die Ausführung mittels:

```
opsi-admin -d method configState_getObjects \
[] '{"configId":"clientconfig.depot.dynamic","objectId":"<client-id>"}'
```

9.14.4 Editieren der Depoteigenschaften

Die Eigenschaften eines Depots werden zum Teil abgefragt, wenn ein opsi-server über den Befehl `opsi-setup --register-depot` als Depot registriert wird (siehe Abschnitt 9.13.2).

Die Depoteigenschaften können Sie nachträglich editieren. Dies geht sowohl im opsi Management Interface als auch auf der Kommandozeile.

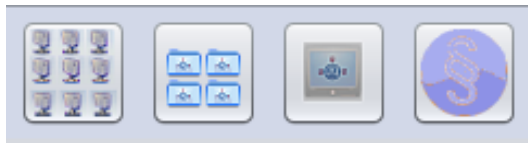


Abbildung 149: Aufruf der Depoteigenschaften (2. Button von links)

Die Depoteigenschaften rufen Sie über den Button *Depoteigenschaften* rechts oben im Managementinterface auf.

Property-Name	Property-Wert
depotLocalUrl	file://opt/pcbin/install
depotRemoteUrl	smb://bonifax/opt_pcbn/install
depotWebdavUrl	webdavs://bonifax.uib.local:4447/depot
description	Autotest-Depot
hardwareAddress	54:52:00:63:99:3b
id	bonifax.uib.local
inventoryNumber	
ipAddress	192.168.1.14
isMasterDepot	true
masterDepotId	null
maxBandwidth	0
networkAddress	192.168.1.0/255.255.255.0
notes	
opsiHostKey	432721195f1ab54a990ab4148bda53ff
repositoryLocalUrl	file://var/lib/opsi/repository
repositoryRemoteUrl	webdavs://bonifax.uib.local:4447/repository
type	Opsiconfigserver

Abbildung 150: Depoteigenschaften im opsi-configed

Auf der Kommandozeile können die Depoteigenschaften ausgegeben werden mit der Methode `host_getObjects`. Hier z.B. für das Depot `dep1.uib.local`.

```
opsi-admin -d method host_getObjects [] '{"id":"dep1.uib.local}"'
```

Dieses Beispiel ergibt folgende Ausgabe:


```
[
  {
    "masterDepotId" : "masterdepot.uib.local",
    "ident" : "dep1.uib.local",
    "networkAddress" : "192.168.101.0/255.255.255.0",
    "description" : "Depot 1 an Master Depot",
    "inventoryNumber" : "",
    "ipAddress" : "192.168.105.1",
    "repositoryRemoteUrl" : "webdavs://dep1.uib.local:4447/repository",
    "depotLocalUrl" : "file:///var/lib/opsi/depot",
    "isMasterDepot" : true,
    "notes" : "",
    "hardwareAddress" : "52:54:00:37:c6:8b",
    "maxBandwidth" : 0,
    "repositoryLocalUrl" : "file:///var/lib/opsi/repository",
    "opsiHostKey" : "6a13da751fe76b9298f4ede127280809",
    "type" : "OpsiDepotserver",
    "id" : "dep1.uib.local",
    "depotWebdavUrl" : "webdavs://dep1.uib.local:4447/depot",
    "depotRemoteUrl" : "smb://dep1/opsi_depot"
  }
]
```

Um die Depoteigenschaften auf der Kommandozeile zu editieren, wird die Ausgabe in eine Datei geschrieben:

```
opsi-admin -d method host_getObjects [] '{"id":"dep1.uib.local"}' \
> /tmp/depot_config.json
```

Die entstandene Datei (/tmp/depot_config.json) kann nun editiert und mit dem folgenden Befehl wieder zurückgeschrieben werden:

```
opsi-admin -d method host_createObjects < /tmp/depot_config.json
```

Die im Rahmen der dynamischen Depotzuweisung wichtigen Depoteigenschaften sind:

- *isMasterDepot*
Muss **true** sein, damit diesem Depot ein Client zugewiesen werden kann. Wird hier **false** eingetragen, so können keine Clients zugewiesen werden, aber die Depots werden trotzdem bei der dynamischen Depotzuweisung verwendet.
- *networkAddress*
Adresse des Netzwerks für den dieses Depot zuständig ist. Die Netzwerkadresse kann nach zwei Notationen angegeben werden:
 - Netzwerk/Maske Beispiel: 192.168.101.0/255.255.255.0
 - Netzwerk/Maskenbits Beispiel: 192.168.101.0/24

Ob die *networkAddress* tatsächlich zur Ermittlung des Depots ausgewertet wird, hängt natürlich von dem im Script übergebenen Algorithmus ab. Der von uib ausgelieferte Default-Algorithmus richtet sich nach diesem Kriterium.

9.14.5 Synchronisation der Depots

Um die Depots synchron zu halten, stellt opsi mehrere Werkzeuge bereit:

- opsi-package-manager
- opsi-product-updater

Der opsi-package-manager kann bei der Installation eines opsi-Paketes durch die Verwendung der Parameter **-d ALL** angewiesen werden, das Paket nicht nur auf dem aktuellen Server sondern auf allen bekannten Depots zu installieren. Beispiel:

```
opsi-package-manager -i opsi-template_1.0-20.opsi -d ALL
```

Durch die Verwendung des Parameters `-D` kann der `opsi-package-manager` angewiesen werden, die Differenzen zwischen Depots aufzulisten. Auch hierbei muss mit der Option `-d` eine Liste von Depots angegeben oder mit `-d ALL` auf alle bekannten Depots verwiesen werden. Beispiel:

```
opsi-package-manager -D -d ALL
```

Der `opsi-package-manager` ist also das Werkzeug, um die Synchronisation auf dem *push* Weg durchzuführen. Dagegen ist das Werkzeug `opsi-product-updater` dafür gedacht, um Depots im *pull* Verfahren zu synchronisieren. Der `opsi-product-updater` kann dazu auf den Depots als cronjob laufen. In der Konfigurationsdatei des `opsi-product-updater` (`/etc/opsi/opsi-product-updater.conf`) ist hierzu in der Sektion `[repository_uib]` der Wert von *active* auf *false* zu setzen und in der Sektion `[repository_master]` der Wert von *active* auf *true* zu setzen. Weiterhin wird in der selben Sektion bei `opsiDepotId` die ID des Depots (FQDN) eingetragen, von dem synchronisiert werden soll. Der `opsi-product-updater` synchronisiert dann gegen die Pakete, die auf dem angegebenen Depot im Verzeichnis `/var/lib/opsi/repository` liegen.



Achtung

Wird auf einem opsi-server ein Paket mit `opsi-package-manager -i` installiert (ohne `-d`), so landet es nicht im repository Verzeichnis. Damit es dorthin kopiert wird, kann man entweder bei der Installation mit `-d` explizit den Namen des Depots angeben oder mit `opsi-package-manager -u <paketname>` den upload in das Repository-Verzeichnis explizit anweisen.

Bitte beachten Sie auch die Beschreibung der beiden Werkzeuge in den entsprechenden Kapiteln des opsi-Handbuchs.

9.14.6 Ablauf

Ist für den Client die Verwendung der dynamischen Depotzuweisung über den Host-Parameter `clientconfig.depot.dynamic` angeschaltet, so lädt dieser über den Webservice vom Server das dort hinterlegte Script und führt es aus.

Das Script, welches der Client verwendet um die Depotauswahl durchzuführen, liegt auf dem Server in der Datei: `/etc/opsi/backendManager/extend.d/70_dynamic_depot.conf`

Der in diesem Script definierten Funktion `selectDepot` werden die folgenden Parameter übergeben:

- `clientConfig`
Informationen zur aktuelle Client-Konfiguration (Hash).
Die Keys des `clientConfig`-Hashes sind momentan:
 - `"clientId"`: opsi-Host-ID des Clients (FQDN)
 - `"ipAddress"`: IP-Adresse des Netzwerk-Schnittstelle zum configserver
 - `"netmask"`: Netzwerk-Maske der Netzwerk-Schnittstelle
 - `"defaultGateway"`: Standard-Gateway
- `masterDepot`
Informationen zum Masterdepot (`opsi-depotserver`-Objekt). Das Masterdepot ist das Depot, dem der Client im Managementinterface zugewiesen ist. Die Attribute des übergebenen `opsi-depotserver`-Objekts entsprechen den Attributen, wie sie von `host_getObjects` (siehe Abschnitt 9.14.4) ausgegeben werden.
- `alternativeDepots`
Informationen zu den alternativen Depots (Liste von `opsi-depotserver`-Objekten). Die Liste der alternativen Depots bestimmt sich aus den Depots, welche bezüglich der gerade benötigten Produkte identisch zum Masterdepot sind.

Auf Basis dieser Informationen kann der Algorithmus nun ein Depot aus der Liste auswählen. Das `opsi-depotserver`-Objekt des zu verwendenden Depots muss von der Funktion zurückgegeben werden. Findet der Algorithmus kein passendes Depot aus der Liste der alternativen Depots oder ist diese leer, so sollte das Masterdepot zurückgegeben werden.

9.14.7 Template des Auswahlscripts

Im Templatescript sind drei Funktionen zur Auswahl eines Depots vorimplementiert.

Die Funktion `depotSelectionAlgorithmByNetworkAddress` überprüft die Netzwerkadressen der übergebenen Depots und wählt jenes Depot aus, bei dem die eigene aktuelle IP-Nummer im Netz des Depots liegt.

Die Funktion `depotSelectionAlgorithmByLatency` sendet ICMP „Echo-Request“-Pakete (ping) an die übergebenen Depots und wählt das Depot mit der niedrigsten Latenzzeit aus.

Die Funktion `depotSelectionAlgorithmByMasterDepotAndLatency` ist gedacht für Umgebungen mit mehreren Master-Depots, denen ihrerseits weitere Slave-Depots zugeordnet sein können. Es wird dabei aus der Menge von Masterdepot des Clients und den zugehörigen Slave-Depots das Depot ausgewählt, welches die geringste Latenzzeit vorweisen kann.

Die Funktion `getDepotSelectionAlgorithm` wird vom Client aufgerufen und gibt den Algorithmus zurück, der für die Auswahl des Depots verwendet werden soll. Ohne Änderung am Templatescript wird hier die Funktion `depotSelectionAlgorithmByNetworkAddress` zurückgegeben.

```
# -*- coding: utf-8 -*-

global depotSelectionAlgorithmByNetworkAddress
depotSelectionAlgorithmByNetworkAddress = \
'''
def selectDepot(clientConfig, masterDepot, alternativeDepots=[]):
    selectedDepot = masterDepot
    logger.info(u"Choosing depot from list of depots:")
    logger.info(u"  Master depot: %s" % masterDepot)
    for alternativeDepot in alternativeDepots:
        logger.info(u"  Alternative depot: %s" % alternativeDepot)
    if alternativeDepots:
        import socket, struct
        # Calculate bitmask of host's ipaddress
        n = clientConfig['ipAddress'].split('.')
        for i in range(4):
            n[i] = forceInt(n[i])
        ip = (n[0] << 24) + (n[1] << 16) + (n[2] << 8) + n[3]

        depots = [ masterDepot ]
        depots.extend(alternativeDepots)
        for depot in depots:
            if not depot.networkAddress:
                logger.warning(u"Network address of depot '%s' not known" % depot)
                continue

            (network, netmask) = depot.networkAddress.split(u'/')
            while (network.count('.') < 3):
                network = network + u'.0'
            if (netmask.find('.') == -1):
                netmask = forceUnicode(socket.inet_ntoa(struct.pack('>I', 0xffffffff ^ (1 << 32 - \
forceInt(netmask)) - 1)))
            while (netmask.count('.') < 3):
                netmask = netmask + u'.0'

            logger.debug(u"Testing if ip %s is part of network %s/%s" % (clientConfig['ipAddress'], network\
, netmask))

            n = network.split('.')
            for i in range(4):
                n[i] = int(n[i])
            network = (n[0] << 24) + (n[1] << 16) + (n[2] << 8) + n[3]
            n = netmask.split('.')
            for i in range(4):
                n[i] = int(n[i])
            netmask = (n[0] << 24) + (n[1] << 16) + (n[2] << 8) + n[3]

            wildcard = netmask ^ 0xFFFFFFFFL
            if (wildcard | ip == wildcard | network):
                logger.notice(u"Choosing depot with networkAddress %s for ip %s" % (depot.\
networkAddress, clientConfig['ipAddress']))
                selectedDepot = depot
```

```

                break
            else:
                logger.info(u"IP %s does not match networkAddress %s of depot %s" % (clientConfig['\
ipAddress'], depot.networkAddress, depot))
                return selectedDepot
    '''

global depotSelectionAlgorithmByLatency
depotSelectionAlgorithmByLatency = \
'''
def selectDepot(clientConfig, masterDepot, alternativeDepots=[]):
    selectedDepot = masterDepot
    logger.info(u"Choosing depot from list of depots:")
    logger.info(u"  Master depot: %s" % masterDepot)
    for alternativeDepot in alternativeDepots:
        logger.info(u"  Alternative depot: %s" % alternativeDepot)
    if alternativeDepots:
        from OPSI.Util.Ping import ping
        from OPSI.Util.HTTP import urlsplit
        depots = [ masterDepot ]
        depots.extend(alternativeDepots)
        latency = {}
        for depot in depots:
            if not depot.repositoryRemoteUrl:
                continue

            try:
                (scheme, host, port, baseurl, username, password) = urlsplit(depot.repositoryRemoteUrl)
                latency[depot] = ping(host)
                logger.info(u"Latency of depot %s: %0.3f ms" % (depot, latency[depot]*1000))
            except Exception, e:
                logger.warning(e)

        if latency:
            minValue = 1000
            for (depot, value) in latency.items():
                if (value < minValue):
                    minValue = value
                    selectedDepot = depot
            logger.notice(u"Choosing depot %s with minimum latency %0.3f ms" % (selectedDepot, minValue\
*1000))
    return selectedDepot
'''

global depotSelectionAlgorithmByMasterDepotAndLatency
depotSelectionAlgorithmByMasterDepotAndLatency = \
'''
def selectDepot(clientConfig, masterDepot, alternativeDepots=[]):
    def getLatencyInformation(depots):
        from OPSI.Util.Ping import ping
        from OPSI.Util.HTTP import urlsplit

        latency = {}
        for depot in depots:
            if not depot.repositoryRemoteUrl:
                continue

            try:
                (scheme, host, port, baseurl, username, password) = urlsplit(depot.repositoryRemoteUrl)
                latency[depot] = ping(host)

                if latency[depot]:
                    logger.info(u"Latency of depot %s: %0.3f ms" % (depot, latency[depot]*1000))
                else:
                    logger.info(u"Latency of depot %s: N/A" % depot)
            except Exception, e:
                logger.warning(e)

        return latency

    def getDepotWithLowestLatency(latency):

```

```

        selectedDepot = None
        if latency:
            minValue = 1000
            for (depot, value) in latency.items():
                if not value:
                    continue
                if (value < minValue):
                    minValue = value
                    selectedDepot = depot
            logger.notice(u"Choosing depot %s with minimum latency %0.3f ms" % (selectedDepot, minValue\
*1000))

        return selectedDepot

    logger.info(u"Choosing depot from list of depots:")
    logger.info(u"  Master depot: %s" % masterDepot)
    for alternativeDepot in alternativeDepots:
        logger.info(u"  Alternative depot: %s" % alternativeDepot)

    if alternativeDepots:
        from collections import defaultdict

        # Mapping of depots to its master.
        # key: Master depot
        # value: All slave depots + master
        depotsByMaster = defaultdict(list)

        allDepots = [masterDepot] + alternativeDepots

        for depot in allDepots:
            if depot.masterDepotId:
                depotsByMaster[depot.masterDepotId].append(depot)
            else:
                depotsByMaster[depot.id].append(depot)

        depotsWithLatency = getLatencyInformation(depotsByMaster[masterDepot.id])
        depotWithLowestLatency = getDepotWithLowestLatency(depotsWithLatency)

        logger.info('Depot with lowest latency: {0}'.format(depotWithLowestLatency))
        if not depotWithLowestLatency:
            logger.info('No depot with lowest latency. Falling back to master depot.')
            depotWithLowestLatency = masterDepot

        return depotWithLowestLatency

    return masterDepot
'''

def getDepotSelectionAlgorithm(self):
    #return depotSelectionAlgorithmByMasterDepotAndLatency
    #return depotSelectionAlgorithmByLatency
    return depotSelectionAlgorithmByNetworkAddress

```

9.14.8 Logging

Wenn die dynamische Depotzuweisung aktiviert ist, so finden sich entsprechende Eintragungen von der Depotauswahl im `opsiclientd.log`. Hier der Log einer gekürzten Beispielsitzung. In dieser ist der Server `bonifax.uib.local` Configserver und Masterdepot für den Client `pctrydetlef.uib.local`. Als Masterserver hat die `bonifax` hier die Netzwerkadresse `192.168.1.0/255.255.255.0`. Als alternatives Depot steht die `stb-40-srv-001.uib.local` zur Verfügung mit der Netzwerkadresse `192.168.2.0/255.255.255.0`. Der Client `pctry4detlef.uib.local` hat die IP-Adresse `192.168.2.109`, liegt also im Netz des alternativen Depots.

```

(...)
[6] [Dec 02 18:25:27] [ opsiclientd ] Connection established to: 192.168.1.14 (HTTP.pyo|421)

```

```
[5] [Dec 02 18:25:28] [ event processing gui_startup ] [ 1] product opsi-client-agent: setup (EventProcessing.\
pyo|446)
[5] [Dec 02 18:25:28] [ event processing gui_startup ] Start processing action requests (EventProcessing.pyo|453)
[5] [Dec 02 18:25:28] [ event processing gui_startup ] Selecting depot for products [u'opsi-client-agent'] (Config.\
pyo|314)
[5] [Dec 02 18:25:28] [ event processing gui_startup ] Selecting depot for products [u'opsi-client-agent'] (__init__\
.pyo|36)
(...)
[6] [Dec 02 18:25:28] [ event processing gui_startup ] Dynamic depot selection enabled (__init__.pyo|78)
(...)
[6] [Dec 02 18:25:28] [ event processing gui_startup ] Master depot for products [u'opsi-client-agent'] is bonifax.uib\
.local (__init__.pyo|106)
[6] [Dec 02 18:25:28] [ event processing gui_startup ] Got alternative depots for products: [u'opsi-client-agent'] (\
__init__.pyo|110)
[6] [Dec 02 18:25:28] [ event processing gui_startup ] 1. alternative depot is stb-40-srv-001.uib.local (__init__.\
pyo|112)
(...)
[6] [Dec 02 18:25:28] [ event processing gui_startup ] Verifying modules file signature (__init__.pyo|129)
[5] [Dec 02 18:25:28] [ event processing gui_startup ] Modules file signature verified (customer: uib GmbH) (\
__init__.pyo|143)
(...)
[6] [Dec 02 18:25:28] [ event processing gui_startup ] Choosing depot from list of depots: (<string>|4)
[6] [Dec 02 18:25:28] [ event processing gui_startup ] Master depot: <OpsiParamServer id 'bonifax.uib.local'> (<\
string>|5)
[6] [Dec 02 18:25:28] [ event processing gui_startup ] Alternative depot: <OpsiParamServer id 'stb-40-srv-001.uib.\
local'> (<string>|7)
[5] [Dec 02 18:25:28] [ event processing gui_startup ] Choosing depot with networkAddress 192.168.2.0/255.255.255.0 \
for ip 192.168.2.109 (<string>|40)
[5] [Dec 02 18:25:28] [ event processing gui_startup ] Selected depot is: <OpsiParamServer id 'stb-40-srv-001.uib.\
local'> (__init__.pyo|171)
(...)
[5] [Dec 02 18:25:28] [ event processing gui_startup ] Mounting depot share smb://stb-40-srv-001/opsi_depot (\
EventProcessing.pyo|415)
(...)
```

9.15 opsi Software On Demand (Kiosk-Mode) (frei)

Siehe auch Kapitel *Nachträgliche Installation des opsi-client-agents*: Abschnitt [6.1.6](#)

9.15.1 Einführung

Das Software-On-Demand-Modul bietet opsi-Administratoren die Möglichkeit, ihren Anwendern eine Auswahl an Produkten zur Verfügung zu stellen. Diese Produkte können vom Anwender, ohne Eingriff von einem Administrator, ausgewählt und die Installation gestartet werden. Diese Dokumentation soll die Funktionsweise des Software-On-Demand-Moduls von opsi erläutern und einen Leitfaden bieten, wie man dieses Modul konfigurieren und pflegen kann.

9.15.2 Vorbedingungen für das Modul

Es sind eine Reihe von Vorbedingungen nötig, um dieses Modul einsetzen zu können. Zunächst werden Produkt-Gruppen benötigt, diese stehen erst ab opsi 4.0 zur Verfügung. Weiterhin werden die Pakete opsi-client-agent und opsi-configed ab Version 4.0.1 benötigt.

Tabelle 26: Benötigte Pakete

opsi-Paket	Version
opsi-client-agent	>=4.0.1-3
opsi-winst	>=4.10.8.12
python-opsi	>=4.0.1-7
opsi-depotserver	>=4.0.1-2
opsi-configed	>=4.0.1.6-1

9.15.3 Konfiguration

Die Konfiguration dieses Moduls basiert auf Produktgruppen und Config-Variablen. Die verwendeten Config-Variablen sind:

- software-on-demand.active
- software-on-demand.product-group-ids

Diese Config-Variablen werden beim Einspielen des opsi-depotserver-Pakets angelegt.

Produktgruppen pflegen

Am komfortabelsten kann man Produktgruppen mit dem opsi-configed anlegen und pflegen. Dafür wechselt man zuerst auf den Tab *Produktkonfiguration*.

Tipp

Seit Version 4.0.1.6 des *opsi-configed* kann man direkt zur Produktkonfiguration wechseln, ohne vorher einen Client auszuwählen.

Die Produktgruppen-Leiste befindet sich über der eigentlichen Produktliste.

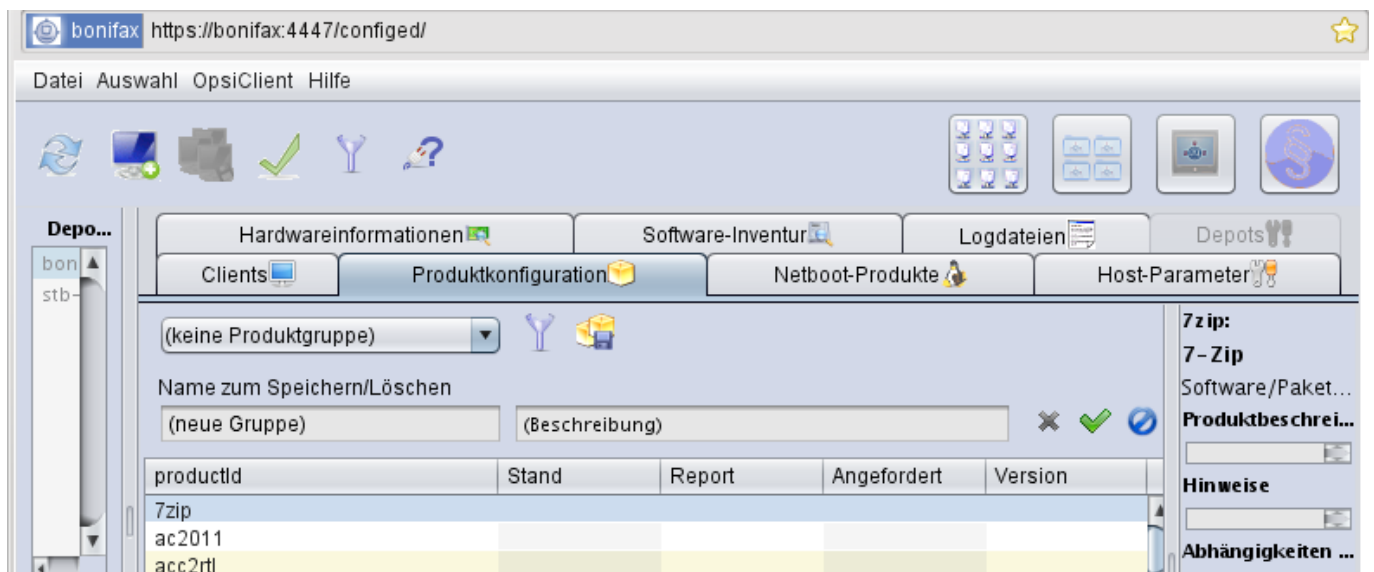


Abbildung 151: Ausschnitt von der Produktgruppen-Leiste

Mit dem Dropdown-Feld kann man Produktgruppen auswählen, um sie zu bearbeiten. Sobald eine Gruppe ausgewählt wurde, werden die dazugehörigen Produkte markiert.

Mit dem zweiten Icon kann man die Filterfunktion ein-, bzw. ausschalten. Bei aktiviertem Filter werden nur die Produkte angezeigt, die der gewählten Produktgruppe zugeordnet sind. Zur Bearbeitung von Produktgruppen aktiviert man die erweiterte Ansicht durch Klick auf das Icon "Paketgruppen mit Diskette". In dieser Ansicht kann eine neue Gruppe, optional Beschreibung, angelegt werden. Durch einen Klick auf den roten Haken, wird die neue Gruppe gespeichert.

Die Zuordnung zur Produktgruppe kann durch Selektieren bzw. Deselektieren von Produkten in der Produktliste bearbeitet werden (Die Taste <STRG> gedrückt halten und Produkte auswählen oder abwählen).

Software-On-Demand-Modul konfigurieren

Mit Hilfe der bereits erwähnten Config-Variablen kann das *SoftwareOnDemand-Modul* flexibel konfiguriert werden. Folgende Tabelle zeigt eine Übersicht der Konfigurationen:

Tabelle 27: Übersicht über die Config-Variablen des SoftwareOnDemand-Moduls

Konfiguration	Beschreibung	Mögliche Werte
software-on-demand.active	Aktiviert bzw. Deaktiviert das Modul.	true/false
software-on-demand.product-group-ids	Produktgruppen mit Produkten, die für Software-On-Demand zur Verfügung stehen sollen.	Liste von Produktgruppen

Es gibt zwei Möglichkeiten diese Konfigurationsobjekte zu verwenden: Systemweit oder pro Client. Die folgenden zwei Unterkapitel gehen auf die zwei verschiedenen Konfigurationsmöglichkeiten näher ein.

Systemweite Konfiguration

Die Einstellungen gelten systemweit als Vorgabe für jeden Client.

Die Konfigurationen können im *opsi-configd* im Modul Servereigenschaften im Tab Host-Parameter bearbeitet werden.

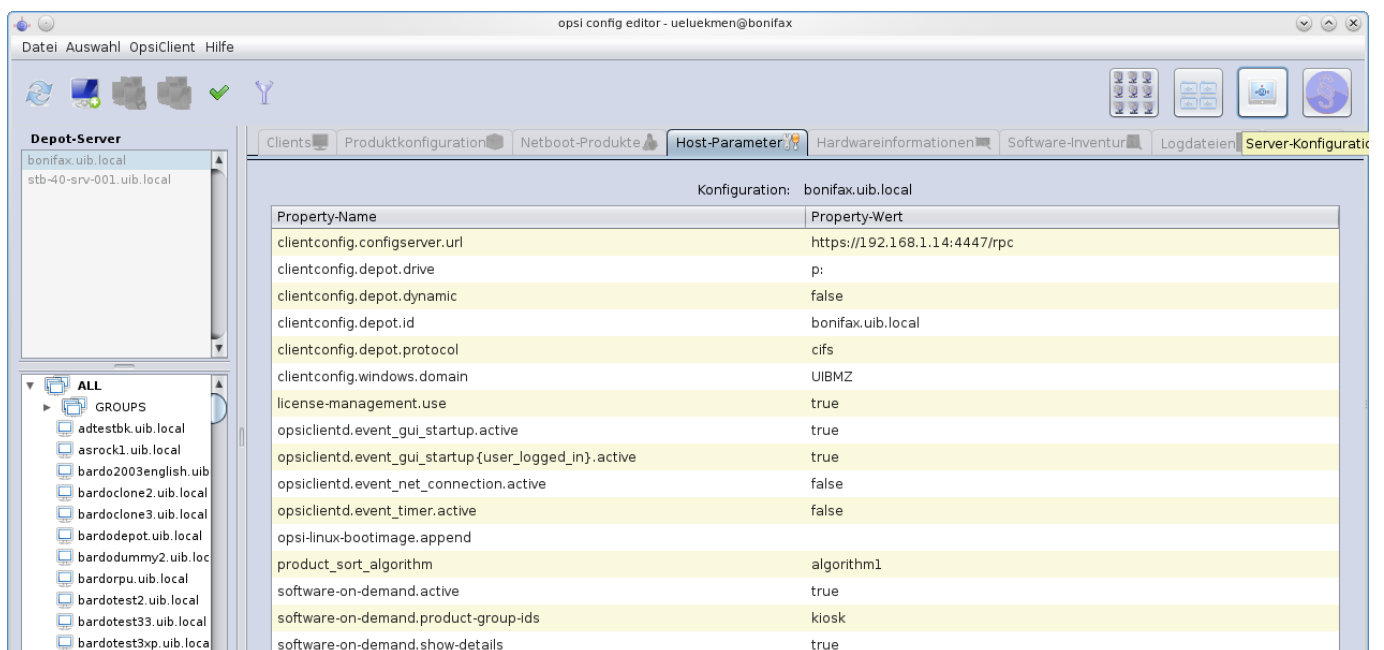


Abbildung 152: Ausschnitt von Serverkonfigurations-Modul des configd

Alternativ kann man die Konfigurationen auf dem Server mittels des folgenden Befehls anpassen:

```
opsi-setup --edit-config-defaults
```

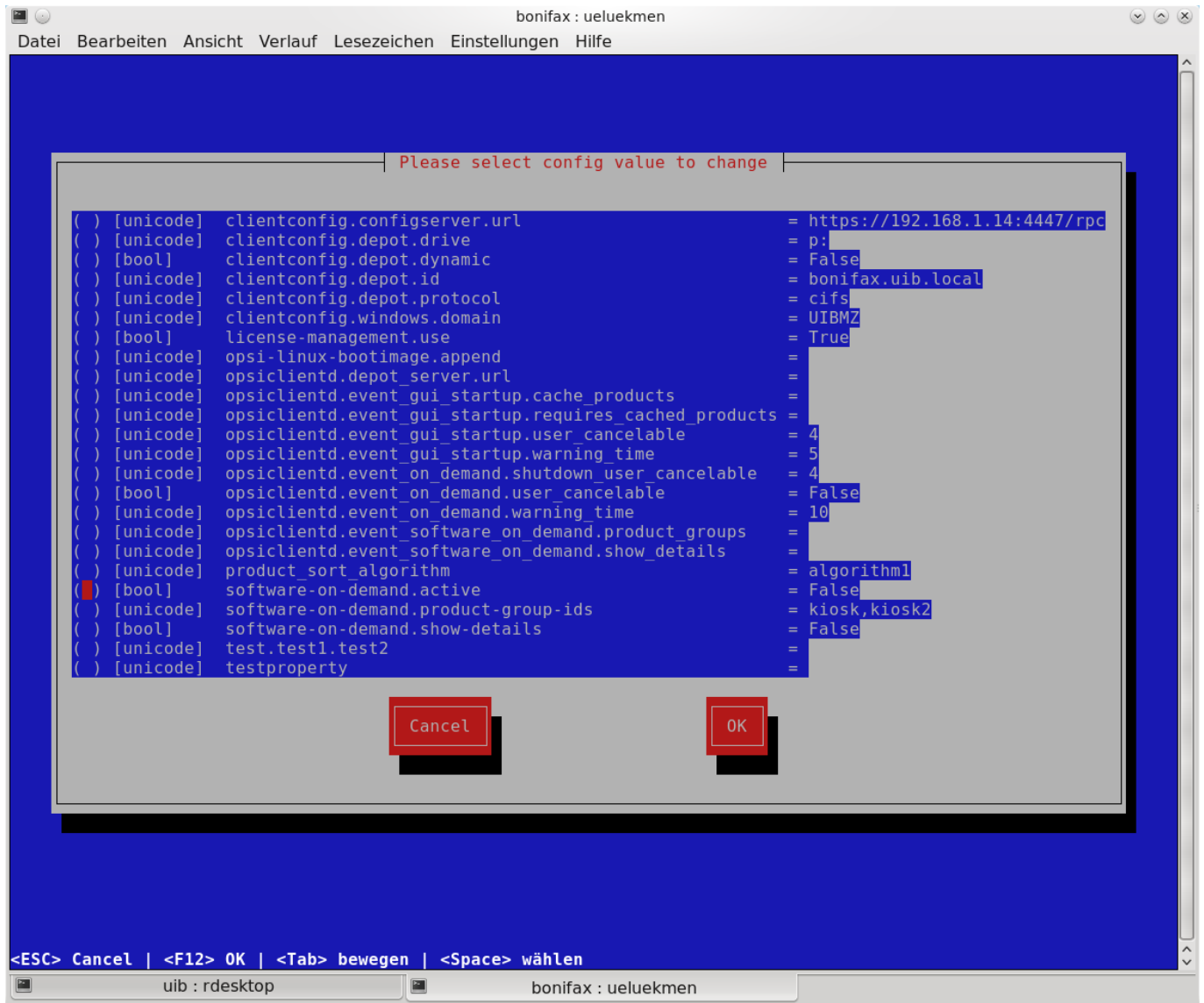



Abbildung 153: Ausschnitt von edit-config-defaults über opsi-setup

Tip

Natürlich ist eine Bearbeitung auch über die opsi-python-API oder über opsi-admin möglich.

Client-spezifische Konfiguration

Die Client-spezifische Konfiguration macht dann Sinn, wenn zum Beispiel nur ein Teil der opsi-Clients Zugriff auf dieses Modul haben soll, oder wenn man verschiedenen Clients unterschiedliche Produktgruppen zur Verfügung stellen will.

Dies erreicht man durch die Konfiguration von Client-spezifischen Host-Parametern. Diese kann man wiederum auf verschiedenen Wegen bearbeiten. Die komfortabelste Möglichkeit diese Konfiguration zu bearbeiten, bietet der opsi-configed. Dafür wählt man im opsi-configed einen oder mehrere Clients (eventuell auch alle Clients einer Clientgruppe) und wechselt auf den Tab Host-Parametern.

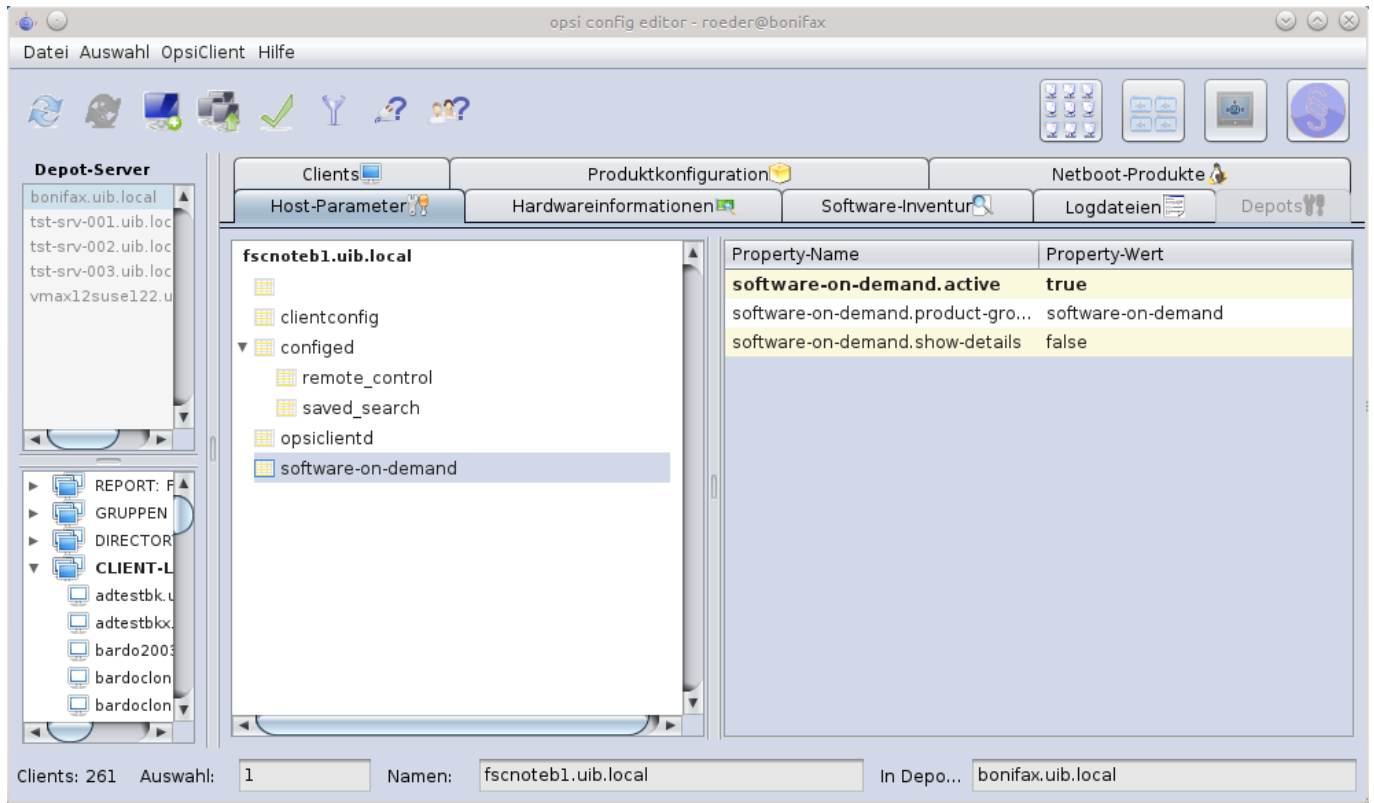


Abbildung 154: Ausschnitt von Host-Parametern

Diese Einstellungen überschreiben die systemweiten Vorgaben.

opsiclientd Event-Konfiguration

Beim Installieren von Produkten über das Software-On-Demand-Modul stehen dem Anwender zwei Möglichkeiten zur Verfügung, die Installation zu starten:

- beim nächsten Systemstart ausführen
- sofort ausführen

Wählt der Benutzer an dieser Stelle die Möglichkeit *beim nächsten Systemstart ausführen*, werden die Produkte nur auf *setup* gesetzt. Wird *sofort ausführen* gewählt, erzeugt der *opsiclientd* ein Event vom Typ *software on demand*. Dieses Event kann, wie jedes andere Event auch, in der *opsiclientd.conf* konfiguriert werden. In der im *opsi-client-agent* enthaltenen *opsiclientd.conf* ist bereits eine Konfiguration enthalten, die angepasst werden kann.

9.15.4 Neue opsi Client Kiosk Anwendung

Mit opsi 4.0.7 wird die bisherige webbasierte Darstellung des Kioskclients (ehemals kofinanzierte Erweiterung "Software on Demand") durch eine Applikation abgelöst. Hintergrund dieses Wechsels sind:

- Beseitigung des Problems das ein selbstsigniertes Zertifikat akzeptiert werden muß.
- Größere Funktionalität des neuen Clients

**Achtung**

Der alte (webseitenbasierte) Kioskclient funktioniert mit dem neuen opsi-client-agent/opsiclientd nicht mehr.

Client Kiosk: Installation

Wenn der opsi-client-agent während der Installation merkt, dass die Konfiguration (Hostparameter): *software-on-demand.active* auf *true* gesetzt wurde, wird automatisch während der Installation auf dem Client ein Startmenü-Eintrag erstellt, über den die Kioskanwendung direkt aufgerufen werden kann. Diesen findet man dann unter: *Start* → *Programme* → *opsi.org* → *software-on-demand*.

Die Installation lässt sich über Properties des Produkts opsi-client-agent modifizieren:

- **kiosk_startmenue_entry**
Steuert den Namen des Startmenü Eintrags.
Default=**software on demand**; Editierbar
- **kiosk_desktop_icon**
Soll ein Desktop-Icon für den Client-Kiosk angelegt werden ?
Default=**false**

Das jeweils verwendete Icon kann durch Ablegen einer **kiosk.ico** Datei unter `/var/lib/opsi/depot/opsi-client-agent/files/opsi/custom/opsiclientkioskskin/` verändert werden.

Client Kiosk: Verwendung

Nach dem Start der Anwendung zeigt sich folgendes Hauptfenster:

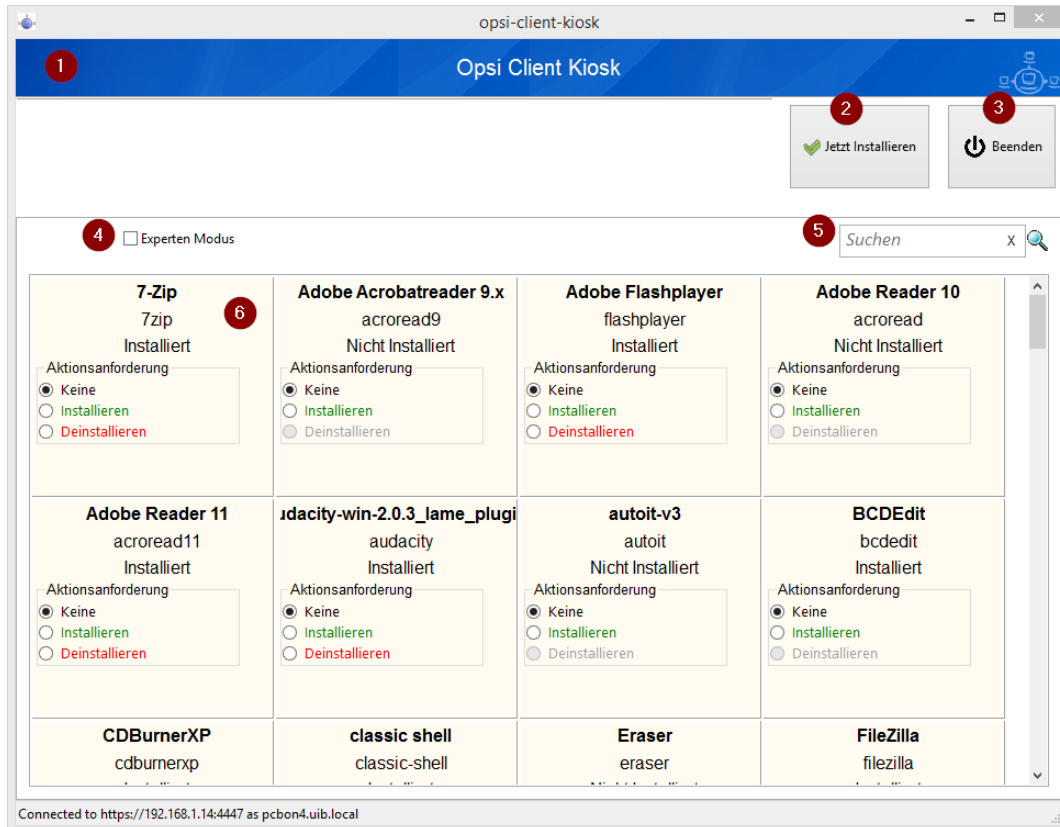


Abbildung 155: opsi-client-kiosk: Hauptfenster mit Kacheln (Default Modus)

Elemente:

1. Headerleiste (Kundenspezifisch anpassbar)
2. Button zum Start der angeforderten Installationen
3. Button zum Beenden des Programms
4. Checkbox zur Aktivierung des Expertenmodus (per Default nicht aktiviert)
5. Suchmaske (Filter Eingabefeld)
6. Kacheln zu den opsi-Paketen

Das Hauptfenster zeigt in dieser Ansicht die freigegebenen Produkte als Kacheln an und mit möglichst wenigen Bedienelementen. Die Produkte werden in der zentralen Bereich (6) angezeigt. Sobald ein Produkt angeklickt ist werden unten Detailinformationen zu diesem Produkt eingeblendet. Durch anklicken der Radiobuttons im Feld *Aktionen anforderung* können Anforderungen gesetzt oder gelöscht werden. Über den Button *Jetzt Installieren* (2) werden die gesetzten Anforderungen an den Server gesendet und die Installation direkt gestartet.

Über das Suchfeld (5) kann nach bestimmten Produkten gesucht werden. Dabei wird in allen Feldern des Produktes gesucht. Über das X im Suchfeld kann das Suchfeld gelöscht werden und damit werden wieder alle Produkte angezeigt. Über die Checkbox *Experten-Modus* (4) können zusätzliche Bedienelemente eingeblendet werden.

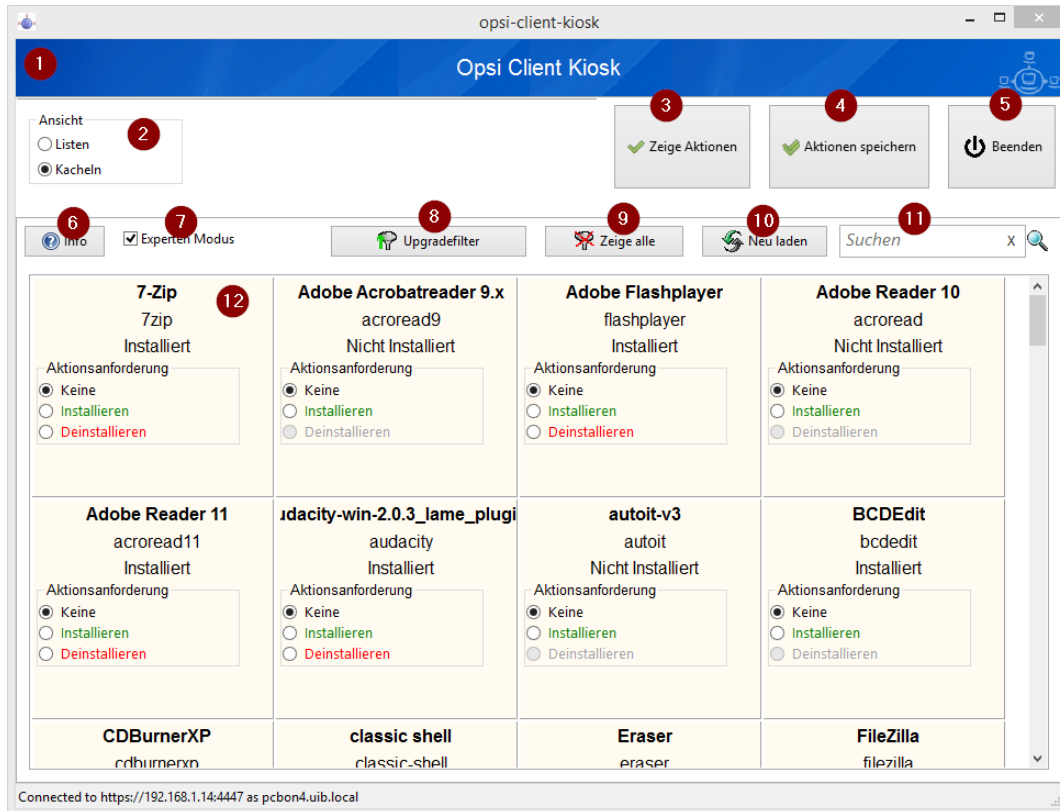


Abbildung 156: opsi-client-kiosk: Hauptfenster mit Kacheln (Experten Modus)

Elemente:

1. Headerleiste (Kundenspezifisch anpassbar)
2. Radiobutton Feld zum Wechsel zwischen Kachelansicht (Default) und Listenansicht
3. Button zum Anzeigen der gesetzten Aktionen
4. Button zum speichern und anzeigen der gesetzten Aktionen
5. Button zum Beenden des Programms
6. Info Button: Version und geladenen Sprache
7. Checkbox zur Aktivierung des Expertenmodus (per Default nicht aktiviert)
8. Nach mögliche Produktupgrades filtern
9. Gesetzten Filter löschen und alle Daten anzeigen
10. Neuladen der Daten (z.B. nachdem Aktionen ausgeführt wurden)
11. Suchmaske (Filter Eingabefeld)
12. Kacheln zu den opsi-Paketen

Das Hauptfenster zeigt in dieser Ansicht die freigegebenen Produkte als Kacheln an und mit zusätzlichen Bedienelementen (Experten-Modus). Die Produkte werden in der zentralen Bereich (6) angezeigt. Sobald ein Produkt angeklickt ist werden unten Detailinformationen zu diesem Produkt eingeblendet. Durch anklicken der Radiobuttons im Feld *Aktionsanforderung* können Anforderungen gesetzt oder gelöscht werden. Über den Button *Zeige*

Aktionen (3) werden die bisher der Anwendung bekannten Aktionen gezeigt aber noch nicht gespeichert. Erst der Button *Aktionen Speichern* (4) sendet die gesetzten Aktionen an den Server. Dieser prüft ob über Produktabhängigkeiten noch weitere Produkte auf setup gesetzt werden müssen. Abschließend wird in einem gesonderten Fenster die Gesamtliste der anstehenden Aktionen angezeigt. Über das Suchfeld (11) kann nach bestimmten Produkten gesucht werden. Dabei wird in allen Feldern des Produktes gesucht. Über das X im Suchfeld kann das Suchfeld gelöscht werden und damit werden wieder alle Produkte angezeigt.

Über die Checkbox *Experten-Modus* (7) sind zusätzliche Bedienungselemente eingeblendet worden.

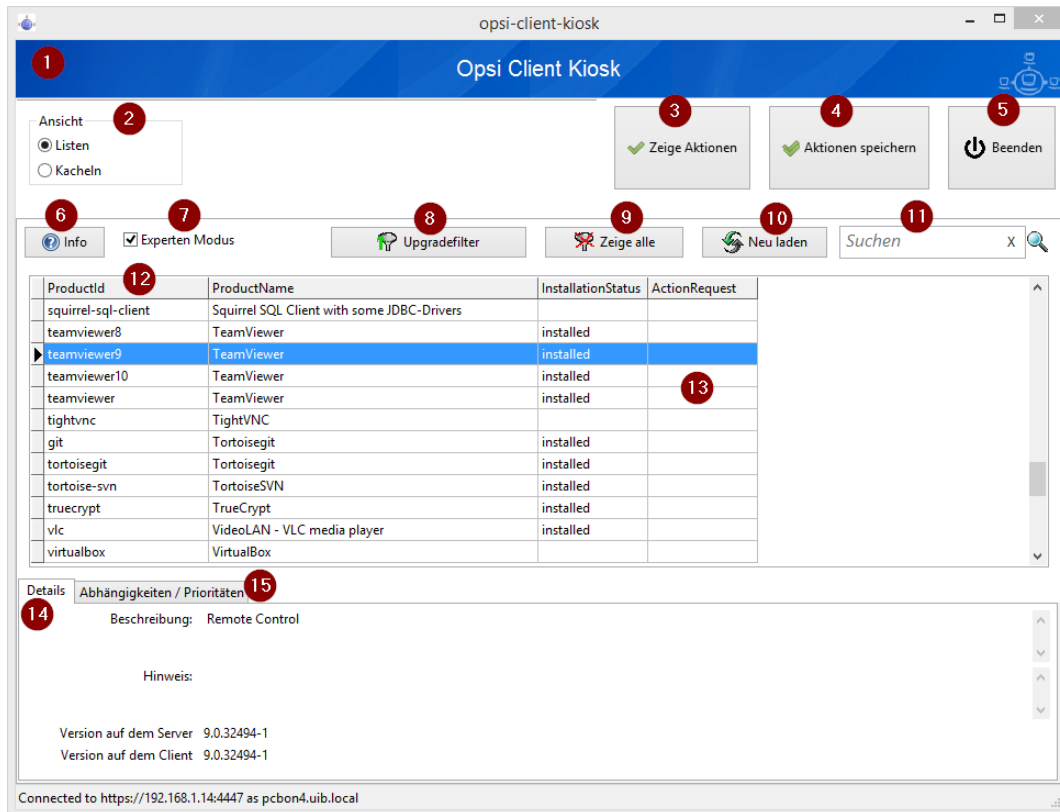


Abbildung 157: opsi-client-kiosk: Hauptfenster mit Listen (Experten Modus)

Elemente:

1. Headerleiste (Kundenspezifisch anpassbar)
2. Radiobutton Feld zum Wechsel zwischen Kachelansicht (Default) und Listenansicht
3. Button zum Anzeigen der gesetzten Aktionen
4. Button zum speichern und anzeigen der gesetzten Aktionen
5. Button zum Beenden des Programms
6. Info Button: Version und geladenen Sprache
7. Checkbox zur Aktivierung des Expertenmodus (per Default nicht aktiviert)
8. Nach mögliche Produktupgrades filtern
9. Gesetzten Filter löschen und alle Daten anzeigen
10. Neuladen der Daten (z.B. nachdem Aktionen ausgeführt wurden)

11. Suchmaske (Filter Eingabefeld)
12. Produktanzeige
13. Spalte zum Setzen der Aktionsanforderungen
14. Tab: Produktdetailinfo: Beschreibung / Hinweis / Versionen
15. Tab: Produktdetailinfo: Abhängigkeiten / Prioritäten

Das Hauptfenster zeigt in dieser Ansicht die freigegebenen Produkte als Liste an. Die Produkte werden in der zentralen Tabelle (12) angezeigt. Sobald ein Produkt angeklickt ist werden unten Detailinformationen zu diesem Produkt eingeblendet (14/15). In der rechten Spalte **ActionRequest** (13) kann eine Aktionsanforderung gesetzt werden. Über den Button *Zeige Aktionen* (3) werden die bisher der Anwendung bekannten Aktionen gezeigt aber noch nicht gespeichert. Erst der Button *Aktionen Speichern* (4) sendet die gesetzten Aktionen an den Server. Dieser prüft ob über Produktabhängigkeiten noch weitere Produkte auf setup gesetzt werden müssen. Abschließend wird in einem gesonderten Fenster die Gesamtliste der anstehenden Aktionen angezeigt. Über das Suchfeld (11) kann nach bestimmten Produkten gesucht werden. Dabei wird in allen Feldern des Produktes gesucht. Über das *X* im Suchfeld kann das Suchfeld gelöscht werden und damit werden wieder alle Produkte angezeigt. Über die Checkbox *Experten-Modus* (7) sind zusätzliche Bedienungselemente eingeblendet worden.

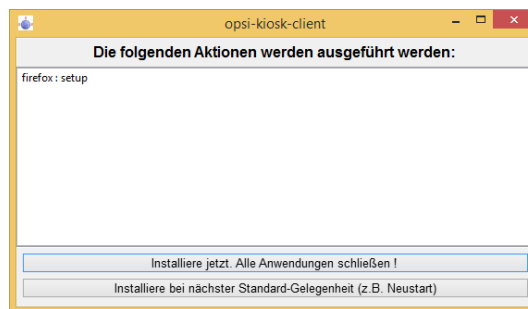


Abbildung 158: *opsi-client-kiosk*: Aktionsbestätigung

Dieses Fenster erscheint nur im *Experten-Modus* und zwar nach dem Betätigen des Buttons *Aktionen Speichern*. In diesem Fenster kann nun über den oberen Button **Installiere jetzt** eine sofortige Installation ausgelöst werden. In diesem Fall ist es schlau möglichst alle Applikationen zu schließen (bzw. zumindest Daten abzuspeichern) da die gestarteten Installationen mit laufenden Anwendungen in Konflikt geraten können. Über den unteren Button **Installiere bei nächster Standard-Gelegenheit** wird das Programm an dieser Stelle beendet und die gespeicherten Aktionen werden später ausgeführt.

Der config *software-on-demand.show-details* aus opsi vor 4.0.7 hat in der Kioskanwendung keinen Einfluß mehr und kann gelöscht werden.

Logging des opsi opsi-client-kiosk Programms:

Das Programm loggt nach `%Appdata%\opsi.org\log`. D.h. in das Verzeichnis `opsi.org\log` im Anwendungsdatenverzeichnis des angemeldeten Users.

Zum Beispiel:

`C:\Users\\AppData\Roaming\opsi.org\log\`

Besonderheiten

Folgende Besonderheiten gelten für das Software On Demand Modul / den opsi Client Kiosk:

- Abhängigkeiten werden automatisch aufgelöst

- Software, die von Software aus der Demand-Gruppe abhängig ist, wird automatisch falls benötigt auf setup gesetzt, ohne Einfluss des Anwenders.
- Software die schon auf setup steht
 - In diesem Fall, wird die Checkbox: *installieren*, schon bei der Übersichtsseite aktiviert.

Client Kiosk: Anpassung an Corporate Identity

Die Anleitung hierfür finden Sie im Kapitel zur Corporate Identity des opsi-client-agent: Abschnitt [6.1.4](#)

9.16 User Profile Management (frei)

9.16.1 Vorbedingungen für die opsi Erweiterung User Profile Management

Dieses Modul war eine [kofinanzierte opsi Erweiterung](#) und ist ab dem 29 Apr 2015 freigegeben. Weitere Details hierzu finden Sie in Abschnitt [9.1](#).

Technische Voraussetzungen sind opsi 4.0.1 mit den Paketständen:

Tabelle 28: Benötigte Pakete

opsi-Paket	Version
opsi-client-agent	>=4.0.1-23
<i>opsi-winst</i>	>=4.11.2.1
python-opsi	>=4.0.1.31-1

Tabelle 29: Benötigte Pakete zur Verwendung ohne Freischaltung

opsi-Paket	Version
opsi-client-agent	>=4.0.5.4-2
<i>opsi-winst</i>	>=4.11.4.17-1



Achtung

Diese Erweiterung funktioniert nicht zusammen mit der WAN-Erweiterung! Schalten Sie auf WAN-Clients bitte das Login-Event nicht an.

9.16.2 Einführung

Der *opsi-winst* verfügt über eine Reihe von speziellen Befehlen um Modifikationen in Profilen vorzunehmen. Diese Arbeiten aber auf den lokalen Profilen und sind beim Einsatz von *Roaming Profiles* (*Servergespeicherte Profile*) weitgehend nutzlos. Mit der opsi Erweiterung *User Profile Management* wird nun eine Möglichkeit geschaffen auch hier Veränderungen an den Profilen vorzunehmen. Dies geschieht in dem beim User Login der *opsi-winst* gestartet wird um spezielle *userLogin.Scripts* auszuführen.

9.16.3 Konzept

Wenn die Profile nicht bei der Installation der Software gleich mit gepatcht werden können, muss zwischen dem *Maschinen Teil* und dem *Profil Teil* der Installation deutlicher unterschieden werden. Die kann sowohl innerhalb eines Scriptes geschehen als auch durch die Auslagerung des *Profil Teils* in ein eigenes Script. Vielerorts passiert dies auch jetzt schon, in dem die *Profil Teile* im Rahmen eines Domain Login Scripts ausgeführt werden.

Je nach Praxis liegen daher die *Profil Teile* von opsi-Produkten als Bestandteil der opsi-scripte zur Installation und Deinstallation vor, als auch als Bestandteil eines Domain Loginscriptes. Ziel dieser Erweiterung ist es, beide Varianten möglichst einfach in den neuen Mechanismus integrieren zu können.

Die Kernkonzepte dieser opsi Erweiterung sind:

- Ausführen spezieller *userLoginScripte* beim Login des users
Im Rahmen des User Logins wird der *opsi-winst* gestartet aber in einem speziellem Modus ausgeführt in dem nur bei den Produkten hinterlegte *userLoginScripte* ausgeführt werden.
- Ausführen der Scripte mit administrativen Rechten aber im Userkontext
Domain Login Scripte werden vom User mit user Rechten ausgeführt. Die opsi *userLoginScripte* werden vom *opsi-winst* ausgeführt, welcher mit administrativen Rechten läuft. Gleichzeitig begibt sich der *opsi-winst* aber in den Kontext des Users der sich eingelogged hat, so dass die Manipulation der Profile mit den selben Befehlen durchgeführt werden kann, wie in einem Domain Loginscript.
- Ausführen der Scripte innerhalb des opsi-service Kontext
Die opsi *userLoginScripts* laufen innerhalb des opsi-service Kontextes und haben so über Scriptkonstanten die Informationen zu Produktnamen, Version und Packageversion die gerade bearbeitet wird. Weiterhin sind die Werte der Produktproperties im Zugriff sowie alle sonstigen Informationen welche eventuell über opsiservicalls abgerufen werden sollen.

Einschränkungen:

- Die *userLoginScripte* werden auch bei der Verwendung der opsi-WAN-Erweiterung nicht aus dem lokalen Cache geladen, sondern online.

9.16.4 Neue und erweiterte *opsi-winst* Funktionen

- Aufrufparameter */allloginscripts*
Wird der *opsi-winst* im opsi-service Kontext mit dem zusätzlichen Parameter */allloginscripts* aufgerufen, so hat das im wesentlichen folgende Auswirkungen:
 - Es werden die Produkte ermittelt welche ein *userLoginScript* haben. Nur für diese Produkte werden die *userLoginScripte* ausgeführt.
 - Es wird der user der sich eingeloggt hat ermittelt und dafür gesorgt, dass die Konstanten zum aktuellen User wie z.B. *%CurrentAppdataDir%* auf die entsprechenden Verzeichnisse des eingelogten users zeigen. Ebenso werden Registry Operationen (**Registry** Sektionen und **GetRegistryString**) welche sich auf HKCU beziehen, so ausgeführt, das die Daten aus dem Registryzweig des Users kommen.
- Aufrufparameter */silent*
Der Aufrufparameter */silent* sorgt dafür, das während der Scriptbearbeitung das Fenster des *opsi-winst* nicht angezeigt wird.
- Funktion **GetScriptMode**
Um innerhalb eines Scriptes zu unterscheiden in welchem Modus das Script gerade ausgeführt wird, liefert die Funktion **GetScriptMode** zwei mögliche Werte zurück:
 - *Machine*
Das Script wird **nicht** als *userLoginScript* ausgeführt (sondern z.B. als setup oder uninstall Script).

- *Login*
Das Script wird als *userLoginScript* ausgeführt.
- Neue primäre Sektion **ProfileActions**
diese neue Sektion kann dazu dienen um Aktionen auf Userprofilen zusammenzufassen. Dabei kann ein Syntax verwendet werden, der es ermöglicht, diese Sektion sowohl als Bestandteil eines normalen Loginscripts als auch als *userLoginScript* zu nutzen. Dazu wird diese primäre Sektion auf unterschiedliche Art ausgewertet, je nachdem ob das script im Machine mode oder Login mode (also als *userLoginScript*) läuft.
 - *Login*
Läuft ein Script als *userLoginScript* und enthält eine Sektion **ProfileActions**, so wird die Scriptbearbeitung bei dieser Sektion gestartet (und nicht bei **Actions**).
 - *Machine*
Läuft ein Script als normales Installationsscript, so kann die Sektion **ProfileActions** ähnlich einer *Sub*-Sektion als Untersektion aufgerufen werden. Für die Abarbeitung dieser Sektion gilt: Für alle *Registry*-Sektions Aufrufe ist implizit */AllNtUserDats* gesetzt. Für alle *Files*-Sektions Aufrufe ist implizit */AllNtUserProfiles* gesetzt. Seit Version 4.11.3.2 gilt: Für alle *Patches*-Sektions Aufrufe ist implizit */AllNtUserProfiles* gesetzt.
- Registry Sektionen
 - Registry Sektionen welche auf *HKCU* bzw. *HKEY_CURRENT_USER* arbeiten, werden im Loginscript Mode so ausgeführt, dass die Änderungen im Zweig des eingeloggten users landen. Entsprechendes gilt für die Funktionen **GetRegistryStringValue***.
 - Registry Sektionen welche im Normalen Modus (*Machine*) mit dem Modifier */AllNtUserDats* aufgerufen werden, dürfen jetzt in der **openkey** Anweisung den Root *HKCU* bzw. *HKEY_CURRENT_USER* enthalten. Dies ermöglicht es, die selbe Registry Sektion in den unterschiedlichen Modi auszuführen.
- Winbatch Sektionen mit **/RunAsLoggedOnUser**
der opsi-winst läuft auch wenn er über das Loginevent gestartet wird im SYSTEM Kontext und nicht im Kontext des users der sich gerade eingeloggt hat. Um einen Prozeß im Kontext dieses Users zu starten kann eine winbatch Sektion mit der Option **/RunAsLoggedOnUser** verwendet werden.
- Vermeidung unnötiger Läufe:
Mit den Befehl **saveVersionToProfile** kann im aktuelle Profil hinterlegt werden, das das userLoginscript zu diesem Produkt in dieser Version gelaufen ist. Mit der Stringfunktion **readVersionFromProfile** bzw. der booleschen Funktion **scriptWasExecutedBefore** kann überprüft werden ob das userLoginScript zu diesem Produkt in dieser Version schon mal zuvor gelaufen ist und eine erneute Ausführung unnötig ist. Dazu liest diese Funktion zunächst einen evtl. vorhandenen Versionsstempel vom Profil ein (wie das mit **readVersionFromProfile** möglich ist) und vergleicht diesen mit der aktuell laufenden Version. Aus dem Vergleich ergibt sich der Rückgabewert (wahr/falsch). Danach werden noch die aktuellen Werte in das Profil zurückgeschrieben (wie das mit **saveVersionToProfile** möglich ist). Somit benötigen Sie nur diese eine Funktion in einer **if** Anweisung, um zu prüfen ob das Script schon mal gelaufen ist.
Weiterhin liefert die Stringlistenfunktion **getProductMap** eine Infomap, aus der entnommen werden kann ob das aktuelle Produkt installiert oder deinstalliert usw. ist.
- Logging
Die Logs von userLoginScripten werden geschrieben nach:
`c:\opsi.org\log\<login user name>_login.log`
Diese Logdateien werden auch an den opsi-server übertragen. Dabei wird eine neue Logdatei an eine existierende angehängt. Der opsi-server sorgt dafür, dass diese Dateien in der Größe beschränkt bleiben (max. 5 MB). Auf dem opsi server liegen diese logs unter `/var/log/opsi/userlogin/<clientid>.log`
Im opsi Managementinterface (opsi-configed) werden diese Logs in einem zusätzliche Untertab *userlogin* in dem Tab *Logdateien* angezeigt.

9.16.5 Beispiele von userLoginScripten

Zunächst zwei Beispiele die so aufgebaut sind, wie sie auch in Domain Loginscripten eingesetzt werden könnten.

Ein sehr einfaches allgemeines Beispiel:

```
[Actions]
requiredWinstVersion >= "4.11.3.2"
Message "Example Profile Patch ...."

Files_profile_copy
Registry_currentuser_set
Patches_profile_ini "%userprofiledir%\opsi-winst-test.ini"

[Files_profile_copy]
copy "%Scriptpath%\profiles\*.*" "%CurrentAppdataDir%\ACME"

[Registry_currentuser_set]
openkey [HKCU\Software\ACME]
set "show_greeting_window" = "no"

[Patches_profile_ini]
add [secdummy] dummy1=add1
```

Ein Beispiel zur Firefoxkonfiguration:

```
[Actions]
requiredWinstVersion >= "4.11.3.2"
Message "Firefox Profile Patch ...."

DefVar $akt_profile_ini$
DefVar $rel_prefs_path$

comment "check for existing profile ..."
Set $akt_profile_ini$ = "%CurrentAppdataDir%\Mozilla\Firefox\profiles.ini"
if FileExists($akt_profile_ini$)
    Set $rel_prefs_path$ = GetValueFromInifile($akt_profile_ini$, "Profile0", "Path", "")
    if FileExists("%CurrentAppdataDir%\Mozilla\Firefox\\"+$rel_prefs_path$)
        comment "We found the profile and will now patch it ....."
    endif
else
    comment "no firefox profile found for user"
endif
```

Als nächstes zeigen wir ein Beispiel welches das erste erweitert um die Möglichkeit Dinge aus dem Profil auch wieder zu entfernen. Je nachdem ob das Produkt auf dem Rechner installiert oder deinstalliert wird ein anderer Scriptteil ausgeführt:

```
[Actions]
requiredWinstVersion >= "4.11.3.2"
Message "Example Profile Patch ...."

if getValue("installationstate", getProductMap) = "installed"
    comment "Product is installed"
    Files_profile_copy
    Registry_currentuser_set
    Patches_profile_ini "%userprofiledir%\opsi-winst-test.ini"
endif
```

```

if getValue("lastactionrequest", getProductMap) = "uninstall"
    comment "Product was uninstalled"
    Files_profile_del
    Registry_currentuser_del
endif

[Files_profile_copy]
copy "%Scriptpath%\profiles\*.*" "%CurrentAppdataDir%\ACME"

[Registry_currentuser_set]
openkey [HKCU\Software\ACME]
set "show_greeting_window" = "no"

[Files_profile_del]
del -s -f "%CurrentAppdataDir%\ACME"
del "%userprofiledir%\opsi-winst-test.ini"

[Patches_profile_ini]
add [secdummy] dummy1=add1

[Registry_currentuser_del]
deletekey [HKCU\Software\ACME]

```

Nun ein Beispiel welches das Setup Skript (setup32.ins und delsub32.ins) nutzt um unnötige Verdopplung des Codes zu vermeiden:

setup32.ins:

```

[Actions]
requiredWinstVersion >= "4.11.3.2"

DefVar $MsiId$
DefVar $UninstallProgram$
DefVar $ProductId$
DefVar $InstallDir$

; -----
; - Please edit the following values -
; -----
Set $ProductId$ = "ACME"
Set $InstallDir$ = "%ProgramFiles32Dir%\ACME"
; -----

if GetScriptMode = "Machine"
    comment "Show product picture"
    ShowBitmap "%ScriptPath%\\" + $ProductId$ + ".png" $ProductId$

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

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

    comment "Start setup program"
    Winbatch_install

    comment "Patch the local Profiles ..."

```

```

Registry_currentuser_set /AllNtUserDats
Files_profile_copy /AllNtUserProfiles
Patches_profile_ini "%userprofiledir%\opsi-winst-test.ini" /AllNtUserProfiles
endif

if GetScriptMode = "Login"
comment "login part"
Files_profile_copy
Registry_currentuser_set
Patches_profile_ini "%userprofiledir%\opsi-winst-test.ini"
endif

[Winbatch_install]
"%ScriptPath%\setup.exe" /sp- /silent /norestart

[Files_profile_copy]
copy "%Scriptpath%\profiles\*.*" "%CurrentProfileDir%\Appdata\ACME"

[Registry_currentuser_set]
openkey [HKCU\Software\ACME]
set "show_greeting_window" = "no"

[Patches_profile_ini]
add [secdummy] dummy1=add1

```

delsub32.ins:

```

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

if GetScriptMode = "Machine"
comment "The machine part ..."
Set $UninstallProgram$ = $InstallDir$ + "\uninstall.exe"
if FileExists($UninstallProgram$)
comment "Uninstall program found, starting uninstall"
Winbatch_uninstall
endif
; does also work since 4.11.2.1
Registry_currentuser_del /AllNtUserDats
Files_profile_del /AllNtUserProfiles
endif

if GetScriptMode = "Login"
comment "The profile part ..."
Files_profile_del
Registry_currentuser_del
endif

[Winbatch_uninstall]
"$UninstallProgram$" /silent /norestart

[Files_profile_del]
del -s -f "%CurrentAppdataDir%\ACME"
del "%userprofiledir%\opsi-winst-test.ini"

[Registry_currentuser_del]
deletekey [HKCU\Software\ACME]

```

Nun ein Beispiel welches eine Variante des vorherigen Beispiels ist. Dabei wird der code durch die Verwendung der neuen primären Sektion `ProfileActions` vereinfacht und das Script ist sowohl als Installationscript als auch als *userLoginScript* verwendbar.

```
[Actions]
requiredWinstVersion >= "4.11.3.2"

DefVar $ProductId$
DefVar $InstallDir$

Set $ProductId$      = "ACME"
Set $InstallDir$     = "%ProgramFiles32Dir%\ACME"

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

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

comment "Start setup program"
Winbatch_install

comment "Patch the local Profiles ..."
ProfileActions

[ProfileActions]
comment "login part"
Files_profile_copy
Registry_currentuser_set
Patches_profile_ini "%userprofiledir%\opsi-winst-test.ini"

[Winbatch_install]
"%ScriptPath%\setup.exe" /sp- /silent /norestart

[Files_profile_copy]
copy "%Scriptpath%\profiles\*.*" "%CurrentProfileDir%\Appdata\ACME"

[Registry_currentuser_set]
openkey [HKCU\Software\ACME]
set "show_greeting_window" = "no"

[Patches_profile_ini]
add [secdummy] dummy1=add1
```

Nun eine Variante, welche sich im Profil merkt ob das Skript für dieses Produkt in dieser Version und diesen User schon mal ausgeführt wurde. Es wird eine Zeile mit den Produkt-Informationen nach Appdata\opsi.org geschrieben.

```
[Actions]
requiredWinstVersion >= "4.11.3.2"
Message "Example Profile Patch ...."

comment "Did we run this script before ? - and set version stamp in profile"
if not (scriptWasExecutedBefore)
    comment "loginscript was not run yet "
    Files_profile_copy
    Registry_currentuser_set
    Patches_profile_ini "%userprofiledir%\opsi-winst-test.ini"
endif
```

```
[Files_profile_copy]
copy "%Scriptpath%\profiles\*.*" "%CurrentAppdataDir%\ACME"

[Registry_currentuser_set]
openkey [HKCU\Software\ACME]
set "show_greeting_window" = "no"

[Patches_profile_ini]
add [secdummy] dummy1=add1
```

9.16.6 Konfiguration

Um die *User Profile Management* Erweiterung zu nutzen muss in der Konfiguration des opsiclientd das Loginevent aktiviert werden. Für dieses Event wird (wenn der entsprechend aktuelle opsi-client-agent auf dem Client installiert ist) der *opsi-winst* mit dem ergänzenden Parameter */alloginscripts* oder */loginscripts* gestartet werden.

- */alloginscripts* bedeutet, das bei einem Login **alle** Loginscripts die dem Server bekannt sind ausgeführt werden, unabhängig ob das entsprechende Produkt dem Client bekannt ist (also installiert ist oder war) oder nicht. Dies ist der Default.
- */loginscripts* bedeutet, das bei einem Login nur die Loginscripts auf einem Client ausgeführt werden, bei denen das entsprechende Produkt dem Client bekannt ist, also installiert ist oder war. (Technisch: bei denen es für diesen Client ein *productOnClient* Objekt gibt). Loginscripts von Produkten die der Client noch nie gesehen hat, werden nicht ausgeführt.

Einen Schalter zur Aktivierung des Loginevents können Sie auf der Kommandozeile wie folgt einrichten: (meist will man zum Test nur einzelne Clients aktivieren, daher serverseitig hier der Wert *false*)

```
opsi-admin -d method config_createBool opsiclientd.event_user_login.active "user_login active" false
```

Als weiterer *opsi-winst* Parameter kann zusätzlich auch noch der Parameter */silent* verwendet werden, welcher die Anzeige des *opsi-winst* Fensters unterbindet.

```
opsi-admin -d method config_createUnicode opsiclientd.event_user_login.action_processor_command "user_login \
action_processor" "%action_processor.command% /sessionid %service_session% /loginscripts /silent" "%\
action_processor.command% /sessionid %service_session% /loginscripts /silent"
```

Die so eingerichteten Einstellungen können Sie im opsi Managementinterface im Tab *Hostparameter* Server- oder Client-spezifisch modifizieren.

9.16.7 Notification

Wenn Sie (wie oben beschrieben) das Loginevent aktiviert haben, so sehen Sie nach jedem Login den *user_login_nofier*:

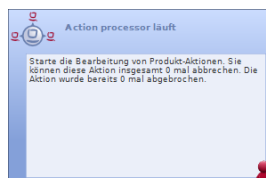


Abbildung 159: User Login Notifier

9.17 opsi Installation beim Shutdown (frei)

9.17.1 Einführung

Standardmäßig wird die Installation von opsi Software-Paketen beim Hochfahren des Clients gestartet. Der Anwender muss dann auf die Beendigung der Software-Installationen warten, bevor er sich am Rechner anmelden kann. Von daher kann es wünschenswert sein, die Software-Installationen vorwiegend beim Runterfahren des Clients durchzuführen.

Das opsi Modul zur Installation beim Shutdown stellt genau diese Funktionalität zur Verfügung. Individuell für bestimmte Clients kann die Installation beim Shutdown aktiviert werden.

9.17.2 Vorbedingungen für die Installation beim Shutdown

Das Modul Installation beim Shutdown kann auf Clients ab **Windows XP** eingesetzt werden. Die erforderliche Grundfunktionalität ist Bestandteil des Paketes *opsi-client-agent*.

Das Paket *opsi-client-agent* muss mindestens die Version 4.0.2.3-2 haben mit einem *opsiclientd* mindestens der Version 4.0.75.

Da die **Kofinanzierungsphase abgeschlossen** ist, wurde das Modul Installation beim Shutdown mit der opsi Version 4.0.5.4-2 freigegeben und ist **nun Bestandteil des freien opsi Systems**. Eine entsprechende *modules*-Datei ist somit ab der aktuellen opsi Version nicht mehr erforderlich. Für ältere *opsi-client-agent* Pakete, d.h. 4.0.2.3-2 bis 4.0.5.4-2, ist noch eine entsprechende *modules*-Datei erforderlich. Weitere Details zu Kofinanzierungs-Projekten finden Sie in Abschnitt 9.1 sowie unter dem Link [kofinanzierte opsi Erweiterungen](#).

Einschränkungen und Besonderheiten bestehen in folgendem Zusammenhang:

- WAN-Erweiterung: das On-Shutdown-Modul ist derzeit nur auf Clients anwendbar, die nicht mit der WAN-Erweiterung betrieben werden. Bei der WAN-Erweiterung ist teilweise die lokale und nicht die serverseitige Konfiguration relevant, dies kollidiert mit der Zustandssteuerung für die Installation beim Shutdown.
- Group Policies: da ein Teil des Mechanismus für die Installation beim Shutdown auf Shutdown-Skripten per Local Group Policy beruht, werden diese Einstellungen eventuell durch übergeordnete Group Policies überschrieben. In diesem Fall sollten die notwendigen Einstellungen in die übergeordneten Group-Policies aufgenommen werden. Siehe hierzu Abschnitt 9.17.4.
- Wenn durch einen Shutdown oder Reboot auf einem Windows-Rechner eine Installation beim Shutdown ausgelöst wurde, lässt die Windows-API zu diesem Zeitpunkt keine Umwandlung eines Shutdown in einen Reboot (oder umgekehrt) mehr zu. Im Falle eines Shutdown werden also Produkte, die einen Reboot beinhalten, erst beim nächsten Hochfahren des Rechners fertig installiert.
- Windows Home-Edition: Windows Home enthält nicht die erforderlichen Group Policy Shutdown-Skript-Mechanismen. Von daher kann die On-Shutdown-Installation auf Windows-Home-Edition nicht eingesetzt werden.
- Windows 2000: Auf Windows 2000 beträgt die maximale Wartezeit für Shutdown-Skripte 10 Minuten, danach wird die Installation durch Windows automatisch abgebrochen. Aus diesem Grund ist das Modul nicht auf Windows 2000 Clients einsetzbar.

9.17.3 Inbetriebnahme der Installation beim Shutdown

Die Grundfunktionalität für die Installation beim Shutdown ist bereits im aktuellen *opsi-client-agent*-Paket enthalten. Seit opsi Version 4.0.5.4-2 ist keine entsprechende *modules*-Datei mehr erforderlich.

Für geeignete Clients (siehe Abschnitt 9.17.2) kann einfach der Produktschalter `on_shutdown_install` des Paketes *opsi-client-agent* auf `on` und der *opsi-client-agent* für diese Clients auf `update` (oder auch `setup`) gesetzt werden.

Weitere Einstellungen sind im Normalfall nicht notwendig.

Außer es werden auf den Clients bereits Group-Policies eingesetzt. Falls es hier zu Kollisionen kommt, sollten die entsprechenden Policy-Einträge nicht automatisch über den *opsi-client-agent* gemacht werden

(`on_shutdown_install_set_policy = off`), sondern händig in die eigene Group-Policy-Verwaltung integriert werden, siehe Abschnitt 9.17.4.

Die Installation beim Shutdown wird zusätzlich zur Installation beim Hochfahren ausgeführt. Dies ist im Normalfall sinnvoll, da somit auch Rechner, die längere Zeit ausgeschaltet waren (z.B. nach Urlaub des Anwenders), vor der Benutzung die neuesten Security-Updates bekommen. Falls gewünscht, kann die Installation beim Hochfahren abgeschaltet werden, siehe Abschnitt 9.17.4. Angefangene Installationen werden aber auf jeden Fall beim Hochfahren des Rechners fortgesetzt, siehe Abschnitt 9.17.4

- Windows unterscheidet beim Herunterfahren systemtechnisch nicht zwischen einem Shutdown und einem Reboot. Die Installation beim Shutdown wird also sowohl beim Shutdown, als auch beim Reboot ausgeführt und es ist nicht möglich, diese beiden Fälle bei der Ausführung eines Skriptes zu unterscheiden. Die Windows-API lässt zu diesem Zeitpunkt weder eine Umwandlung eines Shutdown in einen Reboot (oder umgekehrt) mehr zu, als auch keinen Abbruch des Shutdowns/Reboot. Falls einzelne Software-Pakete eine mehrphasige Installation mit Reboot benötigen, wird die Installation erst beim nächsten Start des Clients fortgesetzt.

9.17.4 Technisches Konzept

Die folgenden Erläuterungen dienen dem besseren Verständnis der technischen Zusammenhänge für spezielle Konfigurationsvarianten sowie der Untersuchung im Fehlerfall. Im Normalfall werden alle erforderlichen Einstellungen vom Paket `opsi-client-agent` durchgeführt.

Überblick

Die Installation beim Shutdown basiert auf dem Zusammenspiel verschiedener System-Komponenten. Ein wesentlicher Bestandteil ist die Nutzung des Windows Shutdown-Skript-Mechanismus per Local Group Policy. Shutdown-Skripte ermöglichen die Durchführung von Tasks genau zu dem Zeitpunkt des Shutdown-Vorgangs, an dem der Benutzer bereits abgemeldet ist und alle Benutzer-Tasks beendet sind, aber noch alle Systemdienste laufen.

Per Shutdown-Skript wird ein opsi-Task ausgeführt, der über den opsi Systemdienst `opsiclientd` eine Installation anstößt und auf deren Beendigung wartet. Erst dann wird das System ganz runter gefahren. Systemtechnisch wird hier nicht zwischen einem Shutdown und einem Reboot unterschieden, so dass die Installation auch bei einem Reboot ausgelöst wird.

Der opsi-Client-Systemdienst `opsiclientd` ist für die Aktionsart `on_shutdown` konfiguriert, die die Installation handhabt. Falls für die Installation Reboots benötigt werden, ist die precondition `installation_pending` für die korrekte Steuerung des Ablaufs zuständig. Falls während der Installation im Shutdown ein Reboot benötigt wird, führt die Precondition `installation_pending` (unabhängig davon, ob `gui_startup` aktiviert ist oder nicht) zu einer direkten Fortführung der Installation beim nächsten Hochfahren des Systems, gegebenenfalls auch mit weiteren Reboots. Im Zustand `installation_pending` wird bei eventuell erforderlichen weiteren Reboots keine Installation beim Shutdown ausgeführt, da ansonsten zwischen der Installation beim Hochfahren und der Installation beim Runterfahren kein Reboot liegen würde. D.h. es wird bis zum Abschluss der aktuellen Installation beim Hochfahren des Systems weiter installiert, aber nicht beim Shutdown, da sonst kein Reboot zwischen den einzelnen Installationsphasen liegen würde.

Im Folgenden werden die beiden Komponenten im Detail beschrieben.

Durchführung per Shutdown-Skript

Über entsprechende Registry-Einträge wird per Local Group Policy beim Herunterfahren des Systems ein Shutdown-Skript ausgeführt, das die Installation anstößt. Die Registry-Einträge entsprechen den Einstellungen, wie sie auch mit dem Group Policy-Editor `gpedit.msc` erzeugt werden können.

So kann man per Group Policy-Editor den Eintrag eines Shutdown-Skriptes erzeugen:

- Richtlinien für Lokaler Computer
- Computerkonfiguration

- Windows-Einstellungen
- Skripts (Start/Herunterfahren)
- Herunterfahren
- Skripts - Hinzufügen - Durchsuchen
- C:\Programme\opsi.org\opsi-client-agent\on_shutdown\doinstall32.cmd (bzw. doinstall64.cmd für 64Bit-Systeme)

Damit das System mit dem Shutdown wartet, bis die Installation vollständig abgeschlossen ist, wird die Wartezeit für Shutdown-Skripte auf unendlich gesetzt (0 Sekunden):

- Richtlinien für Lokaler Computer
- Computerkonfiguration
- Administrative Vorlagen
- System - Skripts
- Maximale Wartezeit für Gruppenrichtlinienskripts
- Einstellung - Aktiviert - Sekunden: 0

Das eingetragene Shutdown-Skript doinstall32.cmd bzw. doinstall64.cmd wechselt das Arbeitsverzeichnis und löst das *on_shutdown*-Event aus:

```
echo Start opsi product installation ...
cd "%ProgramFiles%\opsi.org\opsi-client-agent\on_shutdown"
opsiclientd_event_starter.exe --event=on_shutdown
```

bzw. für 64Bit-Systeme:

```
echo Start opsi product installation ...
cd "%ProgramFiles(x86)%\opsi.org\opsi-client-agent\on_shutdown"
opsiclientd_event_starter.exe --event=on_shutdown
```

Der *opsiclientd_event_starter* wartet auf die Beendigung der Installation, so dass der System Shutdown so lange aufgehalten wird.

Registry-Einträge für die Ausführung des Shutdown-Skripts

Diese Registry-Einträge werden über das Setup des *opsi-client-agent* Paketes gesetzt und führen zur Ausführung des angegebenen Shutdown-Skriptes auf WinXP / 32Bit. Für 64Bit-Systeme ist der Skriptname *doinstall64.cmd* (statt *doinstall32.cmd*).

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\State\Machine\Scripts\Shutdown\0]
"GPO-ID"="LocalGPO"
"SOM-ID"="Local"
"FileSysPath"="C:\\WINDOWS\\System32\\GroupPolicy\\Machine"
"DisplayName"="opsi shutdown install policy"
"GPOName"="opsi shutdown install policy"

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\State\Machine\Scripts\Shutdown\0\0]
"Script"="C:\\Programme\\opsi.org\\opsi-client-agent\\on_shutdown\\doinstall32.cmd"
"Parameters"=""
"ExecTime"=hex(b):00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\System\Scripts\Shutdown\0]
"GPO-ID"="LocalGPO"
"SOM-ID"="Local"
"FileSysPath"="C:\\WINDOWS\\System32\\GroupPolicy\\Machine"
```

```
"DisplayName"="opsi shutdown install policy"
"GPOName"="opsi shutdown install policy"

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\System\Scripts\Shutdown\0\0]
"Script"="C:\Programme\opsi.org\opsi-client-agent\on_shutdown\doinstall32.cmd"
"Parameters"=""
"ExecTime"=hex:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\system]
"MaxGPOScriptWait"=dword:00000000
```

Dies sind die entsprechenden Registry-Einträge für Win6 64Bit (Vista / Win7 / Win8):

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\State\Machine\Scripts\Shutdown\0]
"GPO-ID"="LocalGPO"
"SOM-ID"="Local"
"FileSysPath"="C:\\WINDOWS\\System32\\GroupPolicy\\Machine"
"DisplayName"="opsi shutdown install policy"
"GPOName"="opsi shutdown install policy"

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\State\Machine\Scripts\Shutdown\0\0]
"Script"="C:\\Programme\\opsi.org\\opsi-client-agent\\on_shutdown\\doinstall32.cmd"
"Parameters"=""
"ExecTime"=hex(b):00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\Scripts\Shutdown\0]
"GPO-ID"="LocalGPO"
"SOM-ID"="Local"
"FileSysPath"="C:\\Windows\\System32\\GroupPolicy\\Machine"
"DisplayName"="opsi shutdown install policy"
"GPOName"="opsi shutdown install policy"
"PSScriptOrder"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\Scripts\Shutdown\0\0]
"Script"="C:\\Program Files (x86)\\opsi.org\\opsi-client-agent\\on_shutdown\\doinstall64.cmd"
"Parameters"=""
"IsPowershell"=dword:00000000
"ExecTime"=hex:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\system]
"MaxGPOScriptWait"=dword:00000000
```

Erforderliche Konfiguration des opsiclientd

Der opsi-Client-Systemdienst *opsiclientd* hat für das neue Event *on_shutdown* zusätzliche Standard-Einträge in der Konfigurationsdatei *opsiclientd.conf* bekommen. Hier alle relevanten Einträge:

```
[event_gui_startup]
active = True

[event_gui_startup{installation_pending}]
active = True

[event_on_shutdown]
active = False

[event_on_shutdown{installation_pending}]
active = False

[precondition_installation_pending]
installation_pending = true
```

Die Precondition *installation_pending* zeigt an, ob noch eine Installation im Gange ist. Falls bei Beendigung des Skriptes immer noch der Zustand *installation_pending* auf *true* gesetzt ist, bedeutet dies, dass die aktuelle

Installation einen Reboot benötigt und noch nicht abgeschlossen ist. Im Normalbetrieb ohne Installation bei Shutdown sind die Sektionen für die neue Aktion `event_on_shutdown` deaktiviert.

Für einen Client mit aktivierter Installation bei Shutdown ist dies die erforderliche Konfiguration:

```
[event_gui_startup]
active = True

[event_gui_startup{installation_pending}]
active = True

[event_on_shutdown]
active = True

[event_on_shutdown{installation_pending}]
active = False

[precondition_installation_pending]
installation_pending = true
```

Der einzige Unterschied ist hier also

```
[event_on_shutdown]
active = True.
```

Diese Einstellung wird über den Produktschalter `on_shutdown_install` des Paketes `opsi-client-agent` gesteuert.

Die Precondition `precondition_installation_pending = true` besagt, dass eine angefangene Installation noch nicht beendet ist. Dieser Zustand bleibt über einen oder mehrere Reboots hinweg so lange bestehen, bis die Installation abgeschlossen ist. Wenn die unvollständige Installation einen Reboot benötigt, wird beim nächsten Hochfahren durch die Konfiguration `[event_gui_startup{installation_pending}] active = True` die begonnene Installation beim Hochfahren des Systems fortgesetzt. Diese Einstellung darf nicht verändert werden, da eine angefangene Installation auf jeden Fall beendet werden muss, bevor der Benutzer sich anmelden darf.

Der Eintrag `[event_on_shutdown{installation_pending}] active = False` muss auch in jedem Fall auf `False` bleiben, da bei einer angefangenen Installation ansonsten kein Reboot wäre zwischen den Installationsphasen beim Startup und beim Shutdown.

Sobald die laufende Installation abgeschlossen ist, wird die Precondition `installation_pending = false` und somit auch wieder die Installation im Shutdown aktiv.

Spezielle Konfiguration der Installation bei Shutdown

Im Normalfall ist zur Inbetriebnahme der Installation bei Shutdown, wie in Abschnitt 9.17.3 beschrieben, nichts weiter erforderlich als ein aktuelles `opsi-client-agent` Paket. Für geeignete Clients kann mit dem `opsi-client-agent`-Produktschalter `on_shutdown_install` die Installation beim Shutdown aktiviert werden.

Die Installation beim Hochfahren des Systems bleibt standardmässig auch aktiv. Somit ist gewährleistet, dass auch ein längere Zeit ausgeschalteter Client (z.B. nach dem Urlaub des Benutzers) auf jeden Fall die neuesten Versionsstände bekommt, bevor der Benutzer sich anmelden kann.

Sollte dies nicht erwünscht sein, kann die Installation beim Hochfahren deaktiviert werden. Da die Konfiguration des `opsi-client-agents` auch zentral über den Webservice erfolgen kann (siehe: Abschnitt 6.1.3), ist zu empfehlen, dass folgender `Hostparameter` angelegt wird:

- `opsiclientd.event_gui_startup.active` (boolean, default: true)

Über diesen `Hostparameter` kann dann das `gui_startup`-Event Client-spezifisch aktiviert bzw. deaktiviert werden. Die `Hostparameter` können über den `opsi-configed` oder `opsi-admin` angelegt werden.

Zum Anlegen des `Hostparameter` über `opsi-admin` ist der folgende Befehl auf dem `opsi-configserver` auszuführen:

```
opsi-admin -d method config_createBool opsiclientd.event_gui_startup.active "gui_startup active" true
```

Der Standard-Wert `true` entspricht hierbei dem Wert in der mitgelieferten `opsiclientd.conf`.

Wenn für einen `Install_on_shutdown`-Client die Installation im Startup deaktiviert werden soll, wird der entsprechende *Hostparameter* wie folgt konfiguriert:

- `opsiclientd.event_gui_startup.active: false`

Dies sollte aber nur in begründeten Ausnahmefällen geschehen. Die Einstellungen mit der Zusatzbedingung *installation_pending* sollten unter keinen Umständen geändert werden, es müssen hier immer die default-Werte verwendet werden, um eine korrekte Steuerung des Ablaufs zu gewährleisten.

Zum Setzen des *Hostparameter* über *opsi-admin* ist der folgende Befehl auf dem *opsi-configserver* auszuführen (im Beispiel für einen Client mit der opsi-host-Id `myclient.domain.de`):

```
opsi-admin -d method configState_create opsiclientd.event_gui_startup.active myclient.domain.de false
```

Diese Konfiguration hat zur Folge, dass beim Start des Rechners kein Verbindungsaufbau zum *opsi-configserver* und somit keine Installation stattfindet. Ausser bei einer angefangenen Installation, die durch die Zusatzbedingung *installation_pending* angezeigt wird. In diesem Fall wird durch die Einstellung `event_gui_startup{installation_pending}` beim Systemstart die angefangene Installation weiter fortgesetzt. Wenn ein weiterer Reboot erforderlich ist, wird durch die Einstellung `event_on_shutdown{installation_pending}` (die ebenfalls nicht verändert werden darf) verhindert, dass zusätzlich auch beim Shutdown die Installation weitergeführt wird. Ansonsten wäre kein System-Neustart zwischen den einzelnen Installationsphasen.

Lokale Logdatei für den Fehlerfall

Beim Install-On-Shutdown werden zwei lokale Logdateien angelegt:

- `C:\opsi.org\log\doinstall.log`
- `C:\opsi.org\log\opsiclientd_event_starter.log`

mit normalerweise folgendem Inhalt:

`doinstall.log`:

```
doinstall32.cmd started
Aktuelles Datum: 29.01.2013
```

`opsiclientd_event_starter.log`:

```
[1] [Okt 06 18:49:44:435] opsiclientd_shutdown_starter: version: 4.0.7.0
[5] [Okt 06 18:49:44:435] clientid=pctry4detlef.uib.local
[5] [Okt 06 18:49:44:435] service_url=https://localhost:4441/opsiclientd
[5] [Okt 06 18:49:44:435] service_user=pctry4detlef.uib.local
[5] [Okt 06 18:49:44:450] host_key=***(confidential)***
[5] [Okt 06 18:49:44:450] myevent=on_shutdown
[6] [Okt 06 18:49:44:450] Working with ssl protocol: sslvSSLv23 - auto negotiation
[6] [Okt 06 18:49:45:107] JSON Bench for backend_info "params":[],"id":1} Start: 18:49:44:450 Time: 00:00:00:657
[6] [Okt 06 18:49:45:232] opsidata connected
[5] [Okt 06 18:49:45:232] init Connection done
[6] [Okt 06 18:49:45:232] JSON service request https://localhost:4441/opsiclientd isInstallationPending
[6] [Okt 06 18:49:45:529] JSON Bench for isInstallationPending "params":[],"id":1} Start: 18:49:45:232 Time: \
00:00:00:297
[5] [Okt 06 18:49:45:622] resultstring={"id": 1, "result": false, "error": null}
[5] [Okt 06 18:49:45:622] No installation pending - fine
[6] [Okt 06 18:49:45:622] JSON service request https://localhost:4441/opsiclientd fireEvent
[6] [Okt 06 18:49:45:966] JSON Bench for fireEvent "params":["on_shutdown"],"id":1} Start: 18:49:45:622 Time: \
00:00:00:344
[5] [Okt 06 18:49:46:091] resultstring={"id": 1, "result": null, "error": null}
[5] [Okt 06 18:49:46:091] Succesfull fired event: on_shutdown
[5] [Okt 06 18:49:51:107] calling: isEventRunning,[on_shutdown]
```

```

[6] [Okt 06 18:49:51:107] JSON service request https://localhost:4441/opsiclientd isEventRunning
[6] [Okt 06 18:49:51:357] JSON Bench for isEventRunning "params":["on_shutdown"],"id":1} Start: 18:49:51:107 Time: \
00:00:00:250
[5] [Okt 06 18:49:51:451] resultstring={"id": 1, "result": true, "error": null}
[5] [Okt 06 18:49:56:467] calling: isEventRunning,[on_shutdown]
[6] [Okt 06 18:49:56:467] JSON service request https://localhost:4441/opsiclientd isEventRunning
[6] [Okt 06 18:49:56:935] JSON Bench for isEventRunning "params":["on_shutdown"],"id":1} Start: 18:49:56:467 Time: \
00:00:00:468
[5] [Okt 06 18:49:57:060] resultstring={"id": 1, "result": true, "error": null}
[5] [Okt 06 18:50:02:076] calling: isEventRunning,[on_shutdown]
[6] [Okt 06 18:50:02:076] JSON service request https://localhost:4441/opsiclientd isEventRunning
[6] [Okt 06 18:50:02:545] JSON Bench for isEventRunning "params":["on_shutdown"],"id":1} Start: 18:50:02:076 Time: \
00:00:00:469
[5] [Okt 06 18:50:02:670] resultstring={"id": 1, "result": true, "error": null}
[5] [Okt 06 18:50:07:686] calling: isEventRunning,[on_shutdown]
[6] [Okt 06 18:50:07:686] JSON service request https://localhost:4441/opsiclientd isEventRunning
[6] [Okt 06 18:50:08:186] JSON Bench for isEventRunning "params":["on_shutdown"],"id":1} Start: 18:50:07:686 Time: \
00:00:00:500
[5] [Okt 06 18:50:08:311] resultstring={"id": 1, "result": true, "error": null}
[5] [Okt 06 18:50:13:327] calling: isEventRunning,[on_shutdown]
[6] [Okt 06 18:50:13:327] JSON service request https://localhost:4441/opsiclientd isEventRunning
[6] [Okt 06 18:50:13:624] JSON Bench for isEventRunning "params":["on_shutdown"],"id":1} Start: 18:50:13:327 Time: \
00:00:00:297
[5] [Okt 06 18:50:13:749] resultstring={"id": 1, "result": true, "error": null}
[5] [Okt 06 18:50:18:765] calling: isEventRunning,[on_shutdown]
[6] [Okt 06 18:50:18:765] JSON service request https://localhost:4441/opsiclientd isEventRunning
[6] [Okt 06 18:50:19:030] JSON Bench for isEventRunning "params":["on_shutdown"],"id":1} Start: 18:50:18:765 Time: \
00:00:00:265
[5] [Okt 06 18:50:19:171] resultstring={"id": 1, "result": false, "error": null}
[5] [Okt 06 18:50:19:171] calling: isEventRunning,[on_shutdown{user_logged_in}]
[6] [Okt 06 18:50:19:171] JSON service request https://localhost:4441/opsiclientd isEventRunning
[6] [Okt 06 18:50:19:452] JSON Bench for isEventRunning "params":["on_shutdown{user_logged_in}"],"id":1} Start: \
18:50:19:171 Time: 00:00:00:281
[5] [Okt 06 18:50:19:562] resultstring={"id": 1, "result": false, "error": null}
[5] [Okt 06 18:50:19:562] Task completed

```

oder

```

[1] [Okt 12 18:07:57:352] opsiclientd_shutdown_starter: version: 4.0.7.0
[5] [Okt 12 18:07:57:352] clientId=pctry4detlef.uib.local
[5] [Okt 12 18:07:57:352] service_url=https://localhost:4441/opsiclientd
[5] [Okt 12 18:07:57:352] service_user=pctry4detlef.uib.local
[5] [Okt 12 18:07:57:352] host_key=***(confidential)***
[5] [Okt 12 18:07:57:352] myevent=on_shutdown
[6] [Okt 12 18:07:57:352] Working with ssl protocol: sslvSSLv23 - auto negotiation
[6] [Okt 12 18:07:57:946] JSON Bench for backend_info "params":[],"id":1} Start: 18:07:57:352 Time: 00:00:00:594
[6] [Okt 12 18:07:58:055] opsidata connected
[5] [Okt 12 18:07:58:055] init Connection done
[6] [Okt 12 18:07:58:055] JSON service request https://localhost:4441/opsiclientd isInstallationPending
[6] [Okt 12 18:07:58:290] JSON Bench for isInstallationPending "params":[],"id":1} Start: 18:07:58:055 Time: \
00:00:00:235
[5] [Okt 12 18:07:58:399] resultstring={"id": 1, "result": true, "error": null}
[2] [Okt 12 18:07:58:399] State installation pending detected, do not starting shutdown event.
[2] [Okt 12 18:07:58:399] Terminate called.

```

Diese Logdateien werden jedes Mal neu erzeugt und können im Fehlerfall überprüft werden.

9.18 opsi Feature *SilentInstall* (frei)

Das SilentInstall-Feature bietet opsi-Administratoren die Möglichkeit, Software bei Anwendern zu installieren, ohne dass diese bei Ihrer Arbeit gestört werden. Die folgende Dokumentation beschreibt die Besonderheiten dieser Erweiterung und bietet einen Leitfaden zur Konfiguration der neuen Installationsmethode.

9.18.1 Vorbedingungen für die Silent Installation

Um dieses Feature benutzen zu können, braucht man opsi mindestens in der Version 4.0.3. Hauptsächlich wird der *opsi-client-agent* ab Version 4.0.3.1 benötigt.

9.18.2 Überblick über das SilentInstall-Feature

Die SilentInstall Methode bietet die Möglichkeit eine festgelegte Liste von Produkten zu installieren ohne dass der Anwender seine Arbeit unterbrechen muss. Der große Unterschied zur Installation per onDemand-Event (push Installation) besteht darin, dass bei dieser Installationsmethode nichts angezeigt wird.

Alle Ausgaben werden unterdrückt und auf keinem Desktop angezeigt. Diese Art der Installation birgt natürlich auch Gefahren. Bei einem Problem wie z.B. ein Syntaxfehler im *opsi-winst* Skript, hat man keine Möglichkeit Einfluss auf den Zustand zu nehmen. Der Grund dafür liegt daran, dass eventuell auftauchende Dialogfenster nicht angezeigt werden. Die Folge in diesem Fall wäre, dass der *opsi-winst* nicht beendet wird und damit das Event nicht zum Abschluss kommen kann und eventuell auftretende Events blockiert. Um diesen "Worstcase" zu vermeiden, wird die maximale Installationszeit in Form einer Timeout-Einstellung festgelegt. Dieser Wert muss eventuell angepasst werden, wenn im Vorfeld klar ist, dass die Installation definitiv länger dauert. Näheres dazu wird im Konfigurationsabschnitt dieser Dokumentation beschrieben.

Eine weitere und sehr wichtige Besonderheit dieses Feature ist die fest vorgegebene Liste der Produkte. Es wird zwar Kontakt zum Service hergestellt, aber anders als üblich werden die ActionRequests vom Service ignoriert. Die zu installierende Software wird über eine Konfiguration im *opsi-client-agent* fest vorgegeben. Diese Produkte werden abgearbeitet und müssen dafür nicht extra auf setup gestellt werden. Es wird immer "setup" ausgeführt. Wie in diesem Fall üblich wird nach dem setup-Skript das update-Skript ausgeführt, sofern eines hinterlegt wurde. **Abhängigkeiten werden nicht aufgelöst.** Entweder man sollte komplett auf Pakete mit Abhängigkeiten in diesem Modus verzichten oder man muss dafür sorgen, dass die Produkte inklusive der Abhängigkeiten mit in die Bearbeitungsliste konfiguriert werden sollten. Abschliessend wird der Installationstatus und die Installationslogs zum Service übertragen.

Zusammenfassend wird empfohlen für diesen Modus nur Produkte auszuwählen, die folgende Kriterien erfüllen:

- kleine Pakete oder Installationen
- wenig Last produzieren: Einige Softwareinstallation, unter anderem auch viele MSI Installationen belegen die kompletten Ressourcen des Clients. Somit kann es passieren, dass der Client des Anwenders nicht mehr oder zeitweise sehr träge reagiert.
- in einer festgelegten Zeit installierbar sein: Das Event dieser Erweiterung ist in der Standardkonfiguration mit einem Timeout von 300 Sekunden angelegt. Sollte die Installation in dieser Zeit nicht abgeschlossen sein, wird der *opsi-winst*-Prozess beendet, damit das Event abgeschlossen werden kann.
- Software die einen Reboot bei der Installation benötigt sollte vermieden werden. In der Standardkonfiguration ist dieses Event so konfiguriert, dass angeforderte Reboots nicht ausgeführt werden. Ohne diese Absicherung würde der *opsi-client-agent* ohne Vorwarnung den Client Rebooten. Sollte ein Anwender angemeldet sein, droht Datenverlust. Die Folge daraus könnte sein, dass man eventuell Software im Hintergrund installiert, die in diesem Zustand nicht funktioniert.

In der Standardkonfiguration wird mit diesem Modus swaudit und hwaudit über diese Methode installiert. Die Inventarisierungsprodukte von opsi erfüllen alle oben genannte Kriterien und eignen sich deshalb für diesen Modus. Mit der Standardkonfiguration wird die opsi Hard- und Software-Inventarisierung auf Wunsch ausgeführt, ohne dass die Pakete erst im *opsi-configed* auf setup gestellt werden müssen. Mit dieser Methode kann man bei Bedarf die Informationen in Echtzeit beschaffen. Denkbar wären auch Konfigurationsprodukte, die zum Beispiel Reparaturarbeiten erledigen oder dafür sorgen, dass ein festgelegter Sollzustand der Clienteneinstellungen regelmäßig wiederhergestellt wird.

9.18.3 Auslösen der Silent Installation

Dieses Event wird nicht von alleine ausgelöst wie andere Events. Deshalb gibt es zwei Möglichkeiten dieses Event zu aktivieren.

Die erste Möglichkeit wäre das Event über den opsi-Webservice auszulösen, wie z.B.:

```
opsi-admin -d method hostControl_fireEvent silent_install client.domain.local
```

Dies ermöglicht es auch, diese Schritte in eigenen Skripten unterzubringen. Mit diesen Skripten kann man dieses Feature steuern und z.B.: mit einem at-Job kombinieren, um die Auslösung des Events zu planen.

Als Alternative dazu kann das Event über ein Timer-Event nach einem festgelegten Intervall gestartet werden. In der Standardkonfiguration ist diesem Event ein Intervall von 6 Stunden vorgegeben. Dieser Wert geht davon aus, dass ein Arbeitsplatz-Client im Durchschnitt 8 Stunden eingeschaltet ist. Mit dieser Annahme würde dieses Event jeden Arbeitstag aber nur einmal am Tag ausgeführt werden. Näheres zu der Konfiguration und Aktivierung dieses Events wird im Konfigurationsabschnitt erläutert.

9.18.4 Konfigurationen des opsi-Feature: *SilentInstall*

Im folgenden werden die Standardkonfiguration dieses Features erläutert. Als erstes gibt es folgendes Event in der standard opsiclientd.conf. Das Listing zeigt nur die wichtigen Optionen:

Standard Event SilentInstall:

```
[event_silent_install]
process_shutdown_requests = false
action_processor_productIds = swaudit,hwaudit
action_processor_command = %action_processor.command% /productlist %action_processor_productIds% /silent
action_processor_desktop = winlogon
action_processor_timeout = 300
```

- action_processor_productIds
 - Diese Option ist eine der wichtigen Neuerungen für die Eventsteuerung. Mit dieser Option kann man allen Events, die Produktaktionen ausführen eine Liste von Produkten übergeben. Die Konfiguration für diese Option muss als kommaseparierte Liste angegeben werden.
- process_shutdown_request = false
 - Sorgt dafür, dass ein Reboot, welches vom *opsi-winst* angefordert wurde, nicht ausgeführt wird.
- action_processor_command
 - Hier wird der Aufruf vom *opsi-winst* vorbereitet.
- action_processor_desktop
 - In dieser Option wird beeinflusst auf welchen Desktop die Gui vom *opsi-winst* angezeigt werden soll.
- action_processor_timeout
 - Hier wird der Zeitraum festgelegt, ab wann der *opsi-winst*-Prozess beendet wird

Das zweite Event ist das Timer Event, welches vorgesehen ist, um das Event nach einem festgelegten Intervall auszulösen:

Standard Timer Event für SilentInstall

```
[event_timer_silentinstall]
super = silent_install
type = timer
active = false
interval = 21600
```

- super

- Diese Option beschreibt von welchem Event dieses Timer-Event erbt. Standardmäßig wird vom Event `silent_install` geerbt.
- `type`
 - Diese Option legt fest, dass diese Event-Konfiguration ein Timer-Event ist.
- `active`
 - Standardmäßig ist dieses Event deaktiviert. Um es zu aktivieren muss diese Option auf `true` gestellt werden.
- `interval`
 - Mit dieser Option wird der Intervall festgelegt. Nach Ablauf dieser Zeit wird das Event ausgelöst. Dieser Wert ist Standardmäßig auf 6 Stunden voreingestellt. Diese Zeit sollte wie alle Timerintervalle nicht zu gering gewählt werden, da ansonsten das Event ständig aktiv wird und ggf. andere Aktionen damit blockiert werden. Man sollte das Intervall aber auch nicht zu hoch setzen, da der *opsi-client-agent* über das gesamte Intervall durchlaufen muss, um das Event aus zu lösen. Wenn innerhalb des Intervalls der Client selbst oder der *opsi-client-agent* neu gestartet wird, wird dieses Event nie ausgelöst.

Es ist auch denkbar, das Event `SilentInstall` beim Eintreten eines Systemereignisses auszulösen. Dafür muss zu dem vorhandenen Event noch die Option `wql` konfiguriert werden. Wie man das genau umsetzt kann man beim `event_net_connection` nachschauen. Sollte die `wql` Option eingesetzt werden, wird empfohlen das Event mit `active = false` standardmäßig auszuschalten, damit dieses Event bei Bedarf aktiviert werden kann.

Um das Event wie vorgesehen per Timer zu starten, genügt es eigentlich einen Hostparameter zu setzen. Dazu muss wie gewohnt erst eine Default-Konfiguration angelegt werden. In diesem Fall reicht es aus, das entsprechende Timer-Event zu aktivieren.

Zunächst wird die Standardoption angelegt, im folgenden werden die nötigen *Hostparameter* über *opsi-admin* angelegt. Man kann diese Konfiguration auch über den *opsi-configed* erledigen:

```
opsi-admin -d method config_createBool opsiclientd.event_timer_silentinstall.active "event_timer_silentinstall active" \
false
```

Damit ist für alle Clients dieses Event zunächst deaktiviert. Nun ist es möglich, dieses Event für einzelne Clients aktiv zu setzen:

```
opsi-admin -d method configState_create opsiclientd.event_timer_silentinstall.active silentclient.domain.de true
```

Um die Produkte, die abgearbeitet werden sollen, zu beeinflussen, muss man folgende Einstellungen vornehmen. Wenn man zum Beispiel statt *swaudit* und *hwaudit* das Produkt *firefox* installieren will, sollte man wieder vorgehen wie oben beschrieben. Zunächst muss man einen Defaultwert eintragen:

```
opsi-admin -d method config_createUnicode opsiclientd.event_silent_install.action_processor_productIds "\
event_silent_install productIds" "swaudit,hwaudit"
```

Mit dieser Option wird für alle Clients Standardmäßig, die Produktliste für das Silent Install Event auf *swaudit* und *hwaudit* gesetzt. Um nun diese Einstellung für einen Client zu ändern kann man folgenden Befehl ausführen:

```
opsi-admin -d method configState_create opsiclientd.event_silent_install.action_processor_productIds client.domain.de "\
firefox"
```

Damit ist es auch möglich einem Client eine andere Liste von Produkten zur Verarbeitung zu übergeben.

9.19 opsi Setup Detector (frei)

9.19.1 Einführung

Die grundsätzlichen Schritte bei der Paketierung und Verteilung einer Software sind:

- Analyse des Setups und Erzeugen der Paket-Dateien auf der opsi Workbench
- Packen des opsi Paketes
- Installation des opsi Paketes auf dem opsi Server
- Installation der Software auf den Clients

Der *opsi Setup Detector* ist ein opsi Werkzeug zur Unterstützung der Software-Paketierung. Von seiner grafischen Benutzeroberfläche können alle Schritte von der Auswahl der zu paketierenden Setup-Datei, über das Erstellen der Installationsverzeichnisse auf der opsi Workbench, bis zur Paketierung und Installation auf dem opsi Server durchgeführt werden. Hierzu arbeitet der Setup Detector mit dem Community Projekt *opsi Package Builder* zusammen. Der *opsi Setup Detector* erkennt beim Öffnen einer Setup-Datei, ob es sich um einen bekannten Installer-Typ handelt und ermittelt automatisch die verfügbaren Informationen aus der Setup-Datei. Mit der Zeit wird durch weitere Erfahrungen mit verschiedenen Installer-Typen und Rückmeldungen der Benutzer die Fähigkeit der automatischen Erkennung verschiedener Setup-Typen sicher noch weiter wachsen.

9.19.2 Vorbedingungen für die Benutzung des opsi Setup Detectors

Der *opsi Setup Detector* läuft auf Windows Systemen ab Windows XP.

Er ist Teil der Standard-Repositories von opsi. Er kann mit dem folgenden Befehl installiert werden:

```
opsi-product-updater -i -p opsi-setup-detector
```

Falls die Pakete auch gepackt und auf dem Server installiert werden sollen, wird zusätzlich das Community Projekt opsi Package Builder benötigt, welches auf dem gleichen Windows-Rechner zu installieren ist. Weitere Informationen zum opsi Package Builder finden Sie im opsi Forum bei den opsi Community Projekten: <https://forum.opsi.org/viewforum.php?f=22>

Für die Erzeugung der Pakete ist ein Samba-Share mit Schreibberechtigung auf die opsi Workbench notwendig. Ausserdem ist zu beachten, dass manche Software-Pakete (z.B. manche 64 Bit Software) auch für die Integration ein entsprechendes Windows-System voraussetzen. Das heißt z.B. ein Setup für eine 64bit Windows7 Software kann eventuell nicht auf einem 32bit WindowsXP paketierrt werden. Das hängt im Einzelfall ab vom Verhalten des jeweiligen Setups.

Hier nochmal die Voraussetzungen im Überblick:

- Windows-Rechner ab Windows XP
- opsi package Builder installieren und konfigurieren
- Samba-Share mit Schreibrechten zur opsi Workbench
- *opsi Setup Detector* installieren

9.19.3 Inbetriebnahme des opsi Setup Detectors

Der *opsi Setup Detector* ist ein mit Lazarus entwickeltes Windows-Programm, das zusätzlich einige Hilfsdateien zur Analyse spezieller Setup-Formate sowie die Vorlagen-Dateien zur Erstellung eines neuen opsi Paketes benötigt. Der *opsi Setup Detector* mit seinen Zusatz-Dateien ist als opsi Paket verfügbar. Nach der Installation muss noch der Samba-Share auf die opsi Workbench als Basisverzeichnis eingetragen werden und schon kann mit *Datei öffnen* eine Setup-Datei geöffnet werden, die dann automatisch analysiert wird.

Sprachunterstützung

Beim Start des Programmes wird automatisch ermittelt, unter welcher Sprache das Windows System läuft. Wenn für die Sprache *xx* eine passende Sprachdatei *languages\opsisetupdetector.xx.po* gefunden wird, wird diese verwendet. Von Haus aus unterstützt der *opsi Setup Detector* Deutsch und Englisch. Weitere Sprachen können recht einfach über eine entsprechend zu übersetzende Sprachdatei nachgerüstet werden.

Dateien des opsi Setup Detectors

Folgende Dateien sind im *opsi Setup Detector* Paket enthalten und werden zur Ausführung benötigt:

- *opsisetupdetector.exe* - der *opsi Setup Detector*
- *extractMSI.cmd* - Hilfsdatei zur automatischen Analyse von MSI-Dateien
- *MSIInfo.js* - Hilfsdatei zur automatischen Analyse von MSI-Dateien
- *innounp.exe* - Hilfsdatei zur automatischen Analyse von Inno Setup Paketen
- *innounp.htm* - Hilfsdatei zur automatischen Analyse von Inno Setup Paketen
- *favicon.ico* - opsi Icon
- *languages* - Verzeichnis für die sprachspezifischen Komponenten
- *languages\opsisetupdetector.po* - default Sprachdatei (Englisch)
- *languages\opsisetupdetector.de.po* - Deutsche Sprachdatei
- *languages\Help.en.html* - Englische Hilfe
- *languages\Help.de.html* - Deutsche Hilfe

Zum Erzeugen der opsi Pakete werden verschiedene Vorlagen-Dateien benötigt. Diese werden dann beim Erzeugen eines neuen Paketes in die opsi Workbench kopiert und entsprechend den Einträgen des im *opsi Setup Detector* angezeigten Formulars gepatcht:

- *files2opsi\OPSI* - Vorlagen für die Paketierung-Informationen
- *files2opsi\OPSI\control*
- *files2opsi\OPSI\postinst*
- *files2opsi\OPSI\preinst*
- *files2opsi\CLIENT_DATA* - Vorlagen für die opsiscripte
- *files2opsi\CLIENT_DATA\setup.opsiscript*
- *files2opsi\CLIENT_DATA\uninstall.opsiscript*
- *files2opsi\CLIENT_DATA\delsub.opsiscript*
- *files2opsi\CLIENT_DATA\check_advanced_exitcode.opsiscript*
- *files2opsi\CLIENT_DATA\check_inno_exitcode.opsiscript*
- *files2opsi\CLIENT_DATA\check_installshield_exitcode.opsiscript*
- *files2opsi\CLIENT_DATA\check_installshieldmsi_exitcode.opsiscript*
- *files2opsi\CLIENT_DATA\check_inno_exitcode.opsiscript*
- *files2opsi\CLIENT_DATA\check_msi_exitcode.opsiscript*

9.19.4 Das Menü des opsi Setup Detektors

Das Menü am oberen linken Rand enthält die Punkte

- Datei - (öffnet ein Untermenü, s.u.)
- Hilfe - interne Hilfe aufrufen. Um die enthaltenen Links zu verwenden, kann die Hilfe-Datei `%ProgramFiles-Dir%\opsiSetupDetector\languages\Help.de.html` alternativ auch direkt in einem Browser angezeigt werden.
- Info - Programm-Version anzeigen

Der Menüpunkt *Datei* öffnet ein Untermenü mit:

- *Datei öffnen und analysieren* - Auswählen des zu analysierenden Setup. Dieser Menüpunkt hat die gleiche Funktion wie der *Datei öffnen* Button auf dem *Analysiere* Tab rechts unten und öffnet einen Datei Auswahl Dialog, der alle Dateien anzeigt. Der Datei Auswahl Dialog auf den jeweiligen Installer Tabs zeigt nur Dateien an mit der Endung `.MSI` (MSI Tab) bzw. `.EXE` (alle anderen Installer Tabs).
- *Schreibe Logdatei* - Der Inhalt des Analyze Tab wird als Logdatei geschrieben. Der Name der erzeugten Datei entspricht `%TEMP%\opsitmp\opsiSetupDetector.log` und wird beim Erzeugen in einem Dialog angezeigt. Die Logdatei wird direkt beim Aufruf dieses Kommandos erzeugt und enthält nur die Informationen, die zum Zeitpunkt des Aufrufs im **Analysiere** Tab stehen.
- *Beenden* - beendet den opsi Setup Detektor

9.19.5 Automatische Analyse eines Setup-Paketes

Nach dem Öffnen einer Setup-Datei mit *Datei öffnen* wird diese automatisch analysiert und, falls es sich um einen bekannten Installer-Typ handelt, automatisch auf den entsprechenden Tab gewechselt. Falls der Installer-Typ nicht erkannt wird, bleibt weiterhin das **Analysiere** Tab stehen. Auch wenn der Installer-Typ erkannt wurde, kann jederzeit auf das **Analysiere** Tab zurück gewechselt werden um zu sehen, was bei der automatischen Analyse gefunden wurde.

Bei der Analyse einer MSI-Datei ist bereits durch die Endung MSI bekannt, dass es sich um ein MSI-Paket handelt. Einer EXE-Datei sieht man es aber nicht an der Endung an, mit welchem Installer-Typ sie erstellt wurde. Deshalb wird die EXE-Datei vom *opsi Setup Detector* automatisch nach bestimmten Kennungen durchsucht. Falls die Kennung eines bekannten Installer-Typs gefunden wurde, wird direkt auf den Tab dieses Installers gewechselt. Falls kein bekannter Setup-Typ erkannt wurde, wird die Datei nach bestimmten allgemeinen Stichworten (z.B. "Installer" oder "Setup") durchsucht und die entsprechenden Zeilen werden auf dem **Analysiere** Tab ausgegeben. Darin können auch kryptische Passagen und bestimmte Fehlermeldungen enthalten sein, die das Setup-Programm in bestimmten Fällen, z.B. bei aufgetretenen Fehlern während der Installation, ausgeben möchte, wie z.B.:

```
o@dejDs@s@t@t@du@u@<v@w@w@lx@Hy@x@x{@X|@|@@@L@x@@Inno Setup Setup Data
Das Setup hat einen schwerwiegenden Fehler erkannt und wird abgebrochen
```

Dies besagt aber nur, dass diese Zeichen innerhalb des Setups gefunden wurden und im **Analysiere** Tab angezeigt werden, da sie einem der verwendeten Suchmuster entsprechen. Oben kommt z.B. das Suchmuster "Setup" vor. Bei einem nicht automatisch erkannten Setup-Typ lässt sich in diesen Angaben eventuell ein Hinweis finden, um welchen Installer-Typ es sich handeln könnte. Alle Angaben auf dem **Analysiere** Tab, die eingeleitet werden von:

```
Analyzing: F:\<setup.exe>
```

und abgeschlossen werden mit:

```
default grep finished.
```

entstammen dieser automatischen Analyse und können im Normalfall unbeachtet bleiben. Falls der Setup-Typ erkannt wurde, steht dieser unterhalb von "default grep finished". Dass ein bekannter Setup Typ erkannt wurde merkt man auch daran, dass automatisch auf den entsprechenden Tab gewechselt wurde.

Je nach dem gefundenen Installer-Typ sind im **Analysiere** Tab noch weitere Informationen zu finden zur weiteren Analyse des Setups. Diese werden in den jeweiligen Kapiteln der entsprechenden Installer-Typen behandelt.

Der Inhalt des **Analysiere** Tabs wird vor einer neuen Analyse gelöscht und enthält dementsprechend nur Informationen zu dem aktuellen Setup.

9.19.6 Setup-EXE mit eingebettetem MSI

Bevor auf die einzelnen Installer-Typen eingegangen wird, noch ein paar allgemeine Bemerkungen zu Setup-Dateien mit eingebettetem MSI: derzeit werden Advanced Installer oder InstallShield basierte EXE-Dateien mit eingebettetem MSI erkannt. Das MSI Paket wird automatisch entpackt und die analysierten Eigenschaften im **Advanced+MSI** Tab bzw. **InstallShield+MSI** Tab sowie zusätzlich im **MSI** Tab angezeigt. Eine EXE-Datei mit eingebettetem MSI kann auch mehrere unterschiedliche MSI-Pakete enthalten, z.B. für unterschiedliche Windows-Versionen oder 32bit/64bit. Das Setup entscheidet dann meist je nach den System-Gegebenheiten automatisch, welches MSI-Paket ausgepackt und verwendet wird. In diesem Fall erhält man bei Analyse des Paketes mit dem *opsi Setup Detector* nur das für das aktuelle System, auf dem der *opsi Setup Detector* läuft, passende MSI-Paket. Solche EXE-Dateien haben oft eine Kommandozeilen-Hilfe, die weitere Auskunft über die enthaltenen Pakete und die Aufruf-Möglichkeiten gibt. Auch kann eine Setup-EXE eventuell je nach System oder sonstigen Eigenschaften verschiedene MST-Dateien verwenden. Kurz und gut, eine Setup.EXE kann sich je nach den Umständen unterschiedlich verhalten. In den meisten Fällen braucht man lediglich den MSI Code des eingebetteten MSI für das opsi Deinstallations-Script. Hierbei ist zu beachten, dass ein MSI-Paket für 32bit und 64bit im Normalfall einen unterschiedlichen MSI Codes enthält. Das opsi Deinstallations-Script muss dann dafür sorgen, dass es alle diese MSI Codes kennt und bei der Deinstallation handhabt. Bei solchen Paketen sollte man den *opsi Setup Detector* auf unterschiedlichen Plattformen laufen lassen, um alle MSI-Kennungen zu erfassen. Wenn der *opsi Setup Detector* ein eingebettetes MSI findet, analysiert er dieses und zeigt die Inhalte zusätzlich im **MSI** Tab an. Es kann dann auch das **MSI** Tab als Basis für das opsi Paket genommen werden, aber im Normalfall sollte man besser die EXE-Datei nehmen, da auf diese Weise schneller eine neue Version eingebunden werden kann und die EXE-Datei eventuell auch zusätzliche Dinge tut.

Bei der Analyse wird das MSI und andere Komponenten der EXE-Datei in das Verzeichnis %TEMP%\opsitmp entpackt. Zuvor wird eventueller anderer Inhalt gelöscht, so dass im Verzeichnis %TEMP%\opsitmp immer nur Dateien des aktuell zu analysierenden Setups zu finden sind.

9.19.7 Unterstützte Installer-Typen

Die einzelnen Installer-Typen sind sehr unterschiedlich bezüglich der Standardisierung ihrer Struktur und automatisierten Zugänglichkeit. Bei einem MSI-Paket oder einem auf Inno-Setup basierenden Setup kann eine Menge nützlicher Informationen automatisch detektiert werden, während andere Installer-Typen, wie z.B. NSIS, keine internen Informationen des Setups zugreifbar machen. In vielen Fällen bringt die automatisierte Analyse gute Ergebnisse, aber es kann auch immer wieder Fälle geben, bei denen die automatisierte Vorgehensweise nicht funktioniert oder zumindest händige Nacharbeit erfordert. Mit zunehmenden Erfahrungen beim Einsatz wird der *opsi Setup Detector* seine automatisierten Fähigkeiten mit der Zeit erweitern.

Da MSI-Pakete der wichtigste und am meisten verwendete Installer Typ sind, folgt nach dem **Analysiere** Tab der **MSI** Tab, danach alle weiteren Installer Typ Tabs in alphabetischer Reihenfolge.

Installer-Typ MSI

MSI-Dateien sind das Installations-Datenformat des Microsoft Windows Installers. Setups vom Installer-Typ MSI sind bereits an der Endung MSI als solche zu erkennen. Das Paketformat ist standardisiert, so dass der *opsi Setup Detector* automatisch weitere Informationen aus dem Paket auslesen und im **MSI** Tab, auf den automatisch gewechselt wird, anzeigen kann. Die im **Analysiere** Tab angezeigten Informationen sehen bei einem MSI-Paket z.B. so aus:

```
Analyzing MSI: F:\Setups\MSI\7- zip\7-Zip .msi
```

```
Microsoft (R) Windows Script Host , Version 5.8
```

```
Copyright (C) Microsoft Corporation 1996-2001. Alle Rechte vorbehalten.
```

```

MSI file : F:\uib\setupdetector\Setups\MSI\7-zip\7-Zip.msi
Manufacturer: Igor Pavlov
ProductName: 7-Zip 4.57
ProductVersion: 4.57.00.0
ProductCode: {23170F69-40C1-2701-0457-000001000000}
UpgradeCode: {23170F69-40C1-2701-0000-000004000000}
MSI file size is: 0,9 MB
Estimated required space is: 5,2 MB

get_MSI_info finished

```

Diese Informationen werden aus dem MSI-Paket gelesen und automatisch in die einzelnen Felder des **MSI** Tabs übernommen. Diese Angaben werden benötigt zum Patchen der opsi Installations-Scripte:

- **MSI Datei:** Name der geöffneten MSI Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der aus dem MSI Paket ausgelesene Name der zu installierenden Software. Diesen Namen möchte man wahrscheinlich im Normalfall beibehalten.
- **Produkt Version:** die ausgelesene Version der zu installierenden Software. Auch diese Angabe möchte man wohl in vielen Fällen beibehalten. Sie darf allerdings nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.
- **MSI Produkt Code:** der aus dem Paket ausgelesene Produkt-Code, welcher für das Deinstallations-Script benötigt wird. Vorsicht: ein MSI-Paket für 32bit und 64bit hat normalerweise einen jeweils unterschiedlichen Produkt Code.
- **erforderlich:** dieser vorgeschlagene Wert stammt nicht aus dem MSI-Paket, sondern ist geschätzt als sechsmal die Größe der MSI-Datei und sollte gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das setup.opsiscript, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Größe MSI:** die ermittelte Größe des MSI-Paketes.
- **MST Datei:** wird im gleichen Verzeichnis (also dort, wo die MSI-Datei liegt) eine MST-Datei (Transform-Datei) gefunden, wird diese eingetragen und beim Aufruf des Setup verwendet. Gegebenenfalls kann mit dem *Auswahl*-Button rechts neben dem Feld eine andere MST-Datei gewählt werden oder durch Entfernen des MST Datei-Häkchens vor dem Feld die Verwendung der MST-Datei ausgeschaltet werden.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsi-scripts *license=true* gesetzt.

Im Falle einer EXE-Datei mit eingebettetem MSI wird das MSI Paket automatisch extrahiert und die gefundenen Informationen zusätzlich zum Tab der EXE-Datei (**Advanced+MSI** oder **InstallShield+MSI**) ebenfalls hier im **MSI** Tab angezeigt.

Installer-Typ **Advanced+MSI**

Der Advanced Installer ist ein Werkzeug zur Erstellung von MSI-Paketen, die auch in EXE-Dateien eingebettet sein können. Erkennt der opsi Setup Detektor eine EXE-Datei als ein mit Advanced Installer eingebettetes MSI-Paket, wird die MSI-Datei automatisch aus der EXE-Datei extrahiert und analysiert und die gefundenen Informationen im Tab **Advanced+MSI** angezeigt. Die entsprechenden Informationen werden zusätzlich auch im **MSI** Tab eingetragen. Nähere Angaben zum **MSI** Tab siehe Kapitel Abschnitt [9.19.7](#).

Beim automatischen Auspacken einer eingebetteten MSI-Datei wird im **Analysiere** Tab Folgendes angezeigt:

```

xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Analyzing AdvancedMSI Setup :
F:\Setups\AdvancedMSI\Phase5\phase562install.exe
Analyzing MSI from Setup
F:\Setups\AdvancedMSI\Phase5\phase562install.exe
cmd.exe /C "F:\Setups\AdvancedMSI\Phase5\phase562install.exe" /extract:e:\Temp\opsitmp\
!! PLEASE WAIT !!
!! PLEASE WAIT !! Extracting and analyzing MSI ...
!! PLEASE WAIT !!
e:\Temp\opsitmp\*.msi
Analyzing MSI: e:\Temp\opsitmp\phase5.msi

```

Das MSI wird in das Verzeichnis =%TEMP%\opsitmp= extrahiert und analysiert. Mit dem Ergebnis werden die Felder des **Advanced+MSI** Tab gefüllt:

- **Setup-Datei:** Name der analysierten Setup-Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der aus dem MSI Paket ausgelesene Name der zu installierenden Software. Diesen Namen möchte man wahrscheinlich im Normalfall beibehalten.
- **Produkt Version:** die ausgelesene Version der zu installierenden Software. Auch diese Angabe möchte man wohl in vielen Fällen beibehalten. Sie darf allerdings nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.
- **MSI Produkt Code:** der aus dem Paket ausgelesene Produkt-Code, welcher für das Deinstallations-Script benötigt wird. Vorsicht: ein Advanced-Installer-Paket kann unterschiedliche MSI-Pakete mit unterschiedlichem MSI Produkt Code für z.B. 32bit und 64bit enthalten und es hängt vom System des Paketierungs-PCs ab, welches MSI-Paket ausgepackt und analysiert wird. Meist kann ein 64bit-Paket nicht auf einem 32bit-Rechner entpackt werden. Zur Paketierung einer 64bit-Software wird somit auch ein 64bit-Paketierungs-Rechner gebraucht.
- **erforderlich:** dieser vorgeschlagene Wert ist geschätzt als sechsmal die Größe der Setup-Datei und kann gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das setup.opsiscript, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Dateigröße:** die Größe der Setup-Datei.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsiscrpts *\$LicenseRequired\$=true* gesetzt.

Signatur Advanced+MSI: bei der automatischen Erkennung einer Advanced+MSI-Datei wird in der EXE-Datei gesucht nach dem Marker:

```
name="microsoft.windows.advancedinstallerssetup
```

Installer-Typ Inno Setup

Mit Inno Setup erstellte EXE-Dateien sind weitgehend standardisiert und enthalten eine spezifische Datei mit Namen *install_script.iss*, die automatisch aus der EXE-Datei extrahiert und analysiert werden kann. Mit den ermittelten Informationen werden die Felder des **Inno Setup Tab** gefüllt und dieser angezeigt. Wenn man zurück in den **Analysiere** Tab klickt, sieht man ungefähr folgende Angaben:

```

.....
Analyzing: F:\Setups\Inno\openssl\Win32OpenSSL_Light-1_0_0i.exe
stringsgrep started (verbose:false , skipzero:false )
found string "<description>Inno Setup</description >"
detected Inno Setup
stringsgrep completed (verbose:false , skipzero:false )
.....
Analyzing Inno-Setup:
extract install_script.iss from F:\Setups\Inno\openssl\Win32OpenSSL_Light-1_0_0i.exe to
C:\ProgramData\opsi setup detector\INNO\Win32OpenSSL_Light-1_0_0i\install_script.iss
"E:\Program Files (x86)\opsiSetupDetector\innounp.exe" -x -a -y -d"C:\ProgramData\opsi setu
; Version detected: 5402
#66 install_script.iss
C:\ProgramData\opsi setup detector\INNO\Win32OpenSSL_Light-1_0_0i\install_script.iss
.....
[Setup]
AppName=OpenSSL Light (32-bit)
AppVerName=OpenSSL 1.0.0i Light (32-bit)
AppPublisher=OpenSSL Win32 Installer Team
AppPublisherURL=http://www.openssl.org
AppSupportURL=http://www.slproweb.com
AppUpdatesURL=http://www.slproweb.com/products/Win32OpenSSL.html
DefaultDirName={sd}\OpenSSL-Win32
DefaultGroupName=OpenSSL
OutputBaseFilename=Win32OpenSSL_Light-1_0_0i
Compression=bzip2
.....
Setup file size is: 1,9 MB
Estimated required space is: 11,1 MB
.....
get_inno_info finished
Inno Setup detected

```

Die Informationen aus *install_script.iss* werden automatisch in die einzelnen Felder des **Inno Setup** Tabs übernommen und dann verwendet zum Patchen der opsi Installations-Scripte:

- **Setup-Datei:** Name der analysierten Setup-Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der aus *install_script.iss* ausgelesene Name der zu installierenden Software. Diesen Namen möchte man wahrscheinlich im Normalfall beibehalten.
- **Produkt Version:** die ausgelesene Version der zu installierenden Software. Auch diese Angabe möchte man wohl in vielen Fällen beibehalten. Sie darf allerdings nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.
- **Install. Verzeichnis:** dies ist der Name des Verzeichnisses, in das die Software auf dem Client installiert wird. Dieses wird gebraucht zum Patchen des Deinstallations-Skriptes. Das Installations-Verzeichnis wird aus *install_script.iss* ausgelesen und kann in vielen Fällen direkt verwendet werden. Manchmal kann die Angabe des Installations-Verzeichnisses auch Inno Setup-spezifische Konstanten und Variablen enthalten, die in geschweiften Klammern angegeben sind. Einige der Inno Setup Variablen werden automatisch in opsi Syntax übersetzt, z.B. Es sind aber auch andere Einträge möglich. Hier ist dann händige Nacharbeit angesagt, so dass in diesem Feld nur Bezeichner stehen, die vom opsi Installer (Winst) verstanden werden. In obigem Fall wäre das z.B.: %Systemdrive%\larus Eventuell kann hierzu auch der *install_script.iss* Eintrag *UninstallDisplayName* ausgewertet werden. Eine Beschreibung der Inno Setup Variablen ist in der [Inno Setup Hilfe](#) zu finden.

- **erforderlich:** dieser vorgeschlagene Wert stammt nicht aus der *install_script.iss*, sondern ist geschätzt als sechsmal die Größe der Setup-Datei und sollte gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das *setup.opsiscript*, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Dateigröße:** die ermittelte Größe der Setup-Datei.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsiscripts *\$LicenseRequired=true* gesetzt.

Signatur Inno Setup: bei der automatischen Erkennung einer Inno Setup-Datei wird in der EXE-Datei gesucht nach dem Marker:

```
<description>inno setup</description>
```

Installer-Typ InstallShield

Mit Installshield erstellte EXE-Dateien sind für eine automatisierte Analyse nicht zugänglich. Es kann lediglich automatisch erkannt werden, dass es sich um ein Installshield Setup handelt. Daher werden die vorgeschlagenen Werte aus dem Namen der Setup-Datei erstellt. Wenn diese Datei lediglich *Setup.exe* heißt, wird in den Feldern auch nur *Setup* eingetragen. Hinzu kommt noch, dass es von der jeweiligen Version des verwendeten Installshield abhängt, welche Aufruf-Parameter die Setup-Datei sowie das Deinstallations-Programm akzeptieren. Bei einem Installshield-Paket ist also weitgehend Handarbeit angesagt. Der erste Test könnte darin bestehen, die Aufrufparameter des Setup zu ermitteln, z.B. durch *Setup.exe -?*. Da mit InstallShield erstellte Pakete inzwischen kaum mehr reines InstallShield, sondern zunehmend interne MSI-Pakete enthalten, kommen bei der Paketierung reine InstallShield-Pakete kaum mehr vor, was den Paketierer in Zukunft von einem chronischen Sorgenkind befreit.

Wenn man zurück in den **Analysiere** Tab klickt, sieht man ungefähr folgende Angaben:

```
.....
Analyzing: F:\uib\setupdetector\Setups\InstallShield\viclient\VMware-viclient.exe
stringsgrep started (verbose:false , skipzero:false)
found string "InstallShield"
detected InstallShield Setup
stringsgrep completed (verbose:false , skipzero:false)
.....
Analyzing InstallShield-Setup:
Setup file size is: 32,1 MB
Estimated required space is: 192,6 MB
.....
get_InstallShield_info finished
InstallShield Setup detected
```

Die einzelnen Felder des **InstallShield** Tab sind:

- **Setup-Datei:** Name der analysierten Setup-Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der Name der zu installierenden Software. Dieser muss evtl. händig korrigiert werden
- **Produkt Version:** die aus dem Name der Setup-Datei ermittelte Versionsnummer muss wahrscheinlich händig korrigiert werden. Sie darf nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.

- **Install. Verzeichnis:** dies ist der Name des Verzeichnisses, in das die Software auf dem Client installiert wird. Dieses wird gebraucht zum Patchen des Deinstallations-Skriptes und muss händig eingetragen werden.
- **erforderlich:** dieser vorgeschlagene Wert ist geschätzt als sechsmal die Größe der Setup-Datei und sollte gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das setup.opsiscript, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Dateigröße:** die ermittelte Größe der Setup-Datei.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsiscripts *\$LicenseRequired\$=true* gesetzt.

Signatur InstallShield: bei der automatischen Erkennung einer InstallShield-Datei ohne eingebettetes MSI wird in der EXE-Datei gesucht nach dem Marker:

```
<description>InstallShield.Setup</description>
```

ohne dass der Marker für ein eingebettetes MSI gefunden wird. Falls auch der Marker für ein eingebettetes MSI gefunden wird, handelt es sich um ein **InstallShield+MSI** (s.u.).

Installer-Typ InstallShield+MSI

Mit InstallShield erstellte Software-Pakete enthalten meist intern eine (oder mehrere?) MSI-Dateien. Um die MSI-Datei zu extrahieren, wird ein Batch gestartet, der das Setup startet und zehn Sekunden wartet, welche MSI-Datei beim Starten ausgepackt wird. Dann wird der Setup-Task abgeschlossen. Das extrahierte MSI-Paket wird dann automatisch analysiert und die Informationen in den **InstallShield+MSI** Tab sowie den **MSI** Tab geschrieben. Hier hängt es eventuell auch von der jeweiligen EXE-Datei ab, ob und wenn ja welches MSI-Paket sie auspackt. Eventuell packt sie keines aus, weil das Betriebssystem unpassend ist oder weil erst noch an der GUI eine Frage beantwortet werden muss, bevor ein MSI ausgepackt wird. Falls die automatische Extrahierung nicht klappt, sollte das MSI händig ausgepackt werden und als MSI analysiert werden. Diese Angaben können dann in den **InstallShield+MSI** Tab über tragen werden.

Im **Analysiere** Tab sind typischerweise solche Einträge zu finden:

```
.....
Analyzing: F:\Setups\Installshield -MSI\javavm\jre -6u22-windows-x64.exe
stringsgrep started (verbose:false , skipzero:false)
found strings "Installer,MSI,Database" and "InstallShield"
detected InstallShield+MSI Setup (InstallShield with embedded MSI)
found strings "Installer,MSI,Database" and "InstallShield"
detected InstallShield+MSI Setup (InstallShield with embedded MSI)
stringsgrep completed (verbose:false , skipzero:false)
.....
Analyzing InstallShield+MSI Setup: F:\Setups\Installshield -MSI\javavm\jre -6u22-windows-x64
Analyzing MSI from InstallShield Setup F:\Setups\Installshield -MSI\javavm\jre -6u22-windows
cmd.exe /C E:\Program Files (x86)\opsiSetupDetector\extractMSI.cmd "F:\Setups\Installshield
!! PLEASE WAIT !!
!! PLEASE WAIT !! Extracting and analyzing MSI ...
!! PLEASE WAIT !!
e:\Temp\opsitmp\*.msi
Analyzing MSI: e:\Temp\opsitmp\phase5.msi
cmd.exe /C cscript.exe "E:\Program Files (x86)\opsiSetupDetector\msiinfo.js" "e:\Temp\opsit
.....
Microsoft (R) Windows Script Host, Version 5.8
Copyright (C) Microsoft Corporation 1996-2001. Alle Rechte vorbehalten.
```

```

MSI file: e:\Temp\opsitmp\phase5.msi
Manufacturer: Systemberatung Schommer
ProductName: Phase 5 HTML-Editor
ProductVersion: 5.6.2.3
ProductCode: {20B1B020-DEAE-48D1-9960-D4C3185D758B}
UpgradeCode: {C63B6E47-6A28-44B6-A2C9-2BF084491FAD}
MSI file size is: 0,3 MB
Estimated required space is: 1,7 MB
.....
get_MSI_info finished
Setup file size is: 15,4 MB
Estimated required space is: 92,7 MB
.....
get_InstallShield_info finished
InstallShield+MSI Setup detected

```

Falls die automatische Extrahierung des MSI geklappt hat, werden die gefundenen Werte automatisch in den **InstallShield+MSI** Tab eingetragen:

- **Setup-Datei:** Name der analysierten Setup-Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der aus dem MSI Paket ausgelesene Name der zu installierenden Software. Diesen Namen möchte man wahrscheinlich im Normalfall beibehalten.
- **Produkt Version:** die ausgelesene Version der zu installierenden Software. Auch diese Angabe möchte man wohl im Normalfall beibehalten. Sie darf nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.
- **MSI Produkt Code:** der aus dem Paket ausgelesene Produkt-Code, welcher für das Deinstallations-Script benötigt wird. Vorsicht: ein MSI-Paket kann unterschiedliche MSI Produkt Codes für z.B. 32bit und 64bit enthalten und es hängt vom System des Paketierungs-PCs ab, welches MSI-Paket ausgepackt und analysiert wird. Meist kann ein 64bit-Paket nicht auf einem 32bit-Rechner entpackt werden. Zur Paketierung einer 64bit-Software wird somit auch ein 64bit-Paketierungs-Rechner gebraucht.
- **erforderlich:** dieser vorgeschlagene Wert ist geschätzt als sechsmal die Größe der Setup-Datei und kann gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das setup.opsiscript, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Dateigröße:** die Größe der Setup-Datei.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsiscripts *\$LicenseRequired\$=true* gesetzt.

Signatur InstallShield+MSI: bei der automatischen Erkennung einer InstallShield-Datei mit eingebettetem MSI wird in der EXE-Datei gesucht nach dem Marker für InstallShield und dem Marker für ein eingebettetes MSI:

```

<description>InstallShield.Setup</description>
...
installer , msi , database

```

Installer-Typ NSIS

NSIS Pakete zeichnen sich insbesondere dadurch aus, dass sie kompakt und schnell sind. Allerdings ist es dadurch nicht möglich, dem Paket automatisiert nähere Angaben zu entlocken. Ausser der Erkennung, dass es sich um ein NSIS-Paket handelt, können alle weiteren Informationen lediglich aus dem Namen der Setup-Datei vorgeschlagen werden. Hier ist also händige Nacharbeit angesagt.

Die Einträge im **Analysiere** Tab sehen typischerweise so aus:

```

.....
Analyzing: F:\Setups\NSIS\7z920\7z920.exe
stringsgrep started (verbose:false , skipzero:false)
found string "Nullsoft.NSIS.exehead"
detected NSIS Setup
stringsgrep completed (verbose:false , skipzero:false)
.....
Analyzing NSIS-Setup:
Setup file size is: 1,1 MB
Estimated required space is: 6,4 MB
.....
get_nsis_info finished
NSIS (Nullsoft Install System) detected

```

Die Bedeutung der einzelnen Felder des **NSIS** Tab:

- **Setup-Datei:** Name der analysierten Setup-Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der Name der zu installierenden Software. Dieser muss evtl. händig korrigiert werden
- **Produkt Version:** die aus dem name der Setup-Datei ermittelte Versionsnummer muss wahrscheinlich händig korrigiert werden. Sie darf nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.
- **Install. Verzeichnis:** dies ist der Name des Verzeichnisses, in das die Software auf dem Client installiert wird. Dieses wird gebraucht zum Patchen des Deinstallations-Skriptes und muss händig eingetragen werden.
- **erforderlich:** dieser vorgeschlagene Wert ist geschätzt als sechsmal die Größe der Setup-Datei und sollte gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das setup.opsiscript, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Dateigröße:** die ermittelte Größe der Setup-Datei.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsiscripts *\$LicenseRequired\$=true* gesetzt.

Signatur NSIS: bei der automatischen Erkennung einer NSIS Setup-Datei wird in der EXE-Datei gesucht nach einem der Marker:

```

Nullsoft.NSIS.exehead
...
Nullsoft Install System

```

9.19.8 Erzeugen eines neuen opsi Paketes

Wenn das Setup analysiert und die Felder des entsprechenden Tab gefüllt sind, kann mit dem Button *Erzeuge opsi Paket* (rechts unten) das neue Produkt erzeugt werden.

Achtung:

- als *Basisverzeichnis* muss ein beschreibbarer Share auf die opsi Workbench eingetragen sein
- aus Sicherheitsgründen kann ein opsi Paket nur dann neu erzeugt werden, wenn es noch nicht vorhanden ist. Falls ein bestehendes Paket überschrieben werden soll, muss zuerst das Verzeichnis von der opsi Workbench gelöscht werden.

Links neben dem Button *Erzeuge opsi Paket* befinden sich drei mögliche Auswahl Optionen, die sich auf die Funktion des Buttons beziehen:

- **opsi Paket erzeugen:** falls noch nicht vorhanden, wird das Verzeichnis für das neue opsi Paket erzeugt und die entsprechenden Dateien erstellt und gepatcht. Es wird kein Paket gebaut.
- **opsi Paket erzeugen und opsi Packet Builder starten:** wie oben werden das Verzeichnis und die Dateien erzeugt und dann zum Packen und installieren der *opsi Packet Builder* gestartet. Die Paket Daten werden beim Aufruf in diesen übernommen und das Paket kann gebaut und installiert werden. Danach sollte der *opsi Packet Builder* wieder geschlossen werden, damit er für den nächsten Aufruf neu gestartet werden kann.
- **erzeugen und auto -build -install -quiet:** wie oben werden das Verzeichnis und die Dateien erzeugt und dann zum Packen und installieren der *opsi Packet Builder* gestartet. Die Paket Daten werden beim Aufruf in diesen übernommen und das Paket wird automatisch gebaut und installiert, falls die entsprechenden Häkchen gesetzt sind. Wenn z.B. das Paket nur gepackt, aber nicht installiert werden soll, kann das Häkchen bei **install** entfernt werden. Zusätzlich kann das Häkchen bei **quiet** gesetzt werden, dann werden die angewählten Funktionen automatisch ausgeführt, ohne dass die Benutzeroberfläche des *opsi Packet Builders* angezeigt wird. Ansonsten sollte der *opsi Packet Builder* dann wieder geschlossen werden, damit er für den nächsten Aufruf neu gestartet werden kann.

Zu Installation, Konfiguration und Bedienung des Community Projektes *opsi Packet Builder* siehe <https://forum.opsi.org/viewforum.php?f=22>