Aagon

E-BOOK

WINDOWS 11 AUTOMATISIERT IM NETZWERK VERTEILEN

August 2024

www.aagon.com

Inhalt

Über den Autor

Das E-Book wurde erstellt von Wolfgang Sommergut – Fachautor, Berater und Konferenzsprecher zu verschiedenen Themen der IT.

Einleitung

Microsoft verlagert im Zuge seines Modern Management auch das Deployment des Betriebssystems in die Cloud. Neue Geräte erhalten ihre Windows-Installation dabei nach dem Vorbild mobiler Geräte von einem Online-Service, im konkreten Fall von Autopilot.

Ergänzend dazu benötigen Admins noch Intune, um die Autopilot-Profile zu konfigurieren. Der Domain Join erfolgt dann in ein Hybrid Active Directory oder direkt zu Entra ID.

Während Microsoft seine Cloud-basierte Infrastruktur für das Deployment und Management von Windows ausbaut, mustert es die traditionellen on-prem-Tools für die automatisierte Installation des Betriebssystems zunehmend aus.

So unterstützen die Windows Deployment Services (WDS) den einfachen Workflow mit Hilfe der boot.wim aus den Installationsmedien für Windows 11 nicht mehr. Alternativ sollen Admins stattdessen WinPE von einem WDS-Server booten und das OS von dort installieren. Die WDS beschränken sich damit mehr oder weniger auf die Rolle eines PXE-Servers.

Ein ähnliches Schicksal erleidet das Microsoft Deployment Toolkit (MDT), ein beliebtes kostenloses Werkzeug zum Erstellen eines Custom Images und zur Automatisierung der Installation mit Hilfe von Task Sequences. Es verliert zwar keine Funktionen wie die WDS, wird aber nicht mehr weiterentwickelt und für Windows 11 nicht mehr offiziell unterstützt.

Trotz Microsofts Ausrichtung auf die Cloud auch beim OS-Deployment setzen die meisten Admins aber immer noch auf die bewährten on-prem-Verfahren, und sei es nur als Ergänzung zu Autopilot. Diese sind etwas schneller und effizienter, wenn eine beschädigte Windows-Installation durch das erneute Aufspielen eines OS-Image ersetzt werden soll.

Die dafür nötigen Tools liefert Microsoft mit dem Windows ADK aus, wobei Windows PE mittlerweile separat heruntergeladen werden muss. Sie unterstützen traditionell zwei Verfahren für die automatisierte Installation von Windows.

Die erste besteht in der Ausführung des Setup-Programms, das standardmäßig jedoch interaktiv abläuft und zahlreiche Eingaben für die Konfiguration des Systems erfordert. Um diese zu vermeiden, kann man die in den verschiedenen Dialogen abgefragten Einstellungen mit Hilfe einer Antwortdatei mit den gewünschten Werten versehen. Sie lässt sich mit dem Windows System Image Manager (SIM) erzeugen bzw. bearbeiten.

Die alternative Methode besteht darin, eine Musterinstallation als WIM-Datei zu erfassen und mit DISM oder PowerShell auf die Ziel-PCs anzuwenden. In diesem Fall muss man einige Aufgaben, wie etwa das Partitionieren des Laufwerks, selbst per Script erledigen.

Vorbereitungen

Unabhängig davon, für welche Methode man sich entscheidet, muss man einige Tools und Komponenten bereitstellen, die für das Deployment benötigt werden:

- Dazu gehört zuvorderst ein Boot-Image, in der Regel mit Windows PE. Es besteht aber auch die Möglichkeit, ein angepasstes WIM-Archiv wieder in die ursprünglichen Installationsmedien zu übernehmen.
- Will man die Clients nicht von einem physischen Medium wie bespielweise einem USB-Stick, sondern über das Netz booten, dann braucht man dafür einen PXE-Server. In Windows-Umgebungen bieten sich dafür die Deployment Services (WDS) an.
- Wenn man darüber hinaus die Dialoge zur Systemkonfiguration während der OOBE-Phase überspringen und die entsprechenden Werte automatisch zuweisen möchte, dann benötigt man dafür eine Antwortdatei. Benutzer durchlaufen diesen Abschnitt nach dem ersten Anmelden, egal wie Windows 11 installiert wurde.

Boot-fähigen Datenträger mit Windows PE 11 erstellen

Das Preinstallation Environment (WinPE) ist eine Magerversion von Windows, die für das OS-Deployment oder für das Recovery Environment benötigt wird. Microsoft stellt es nicht als ISO bereit, sondern man muss einen boot-fähigen Datenträger selbst erzeugen. Dabei kann man auch Sprachpakete oder Tools hinzufügen.

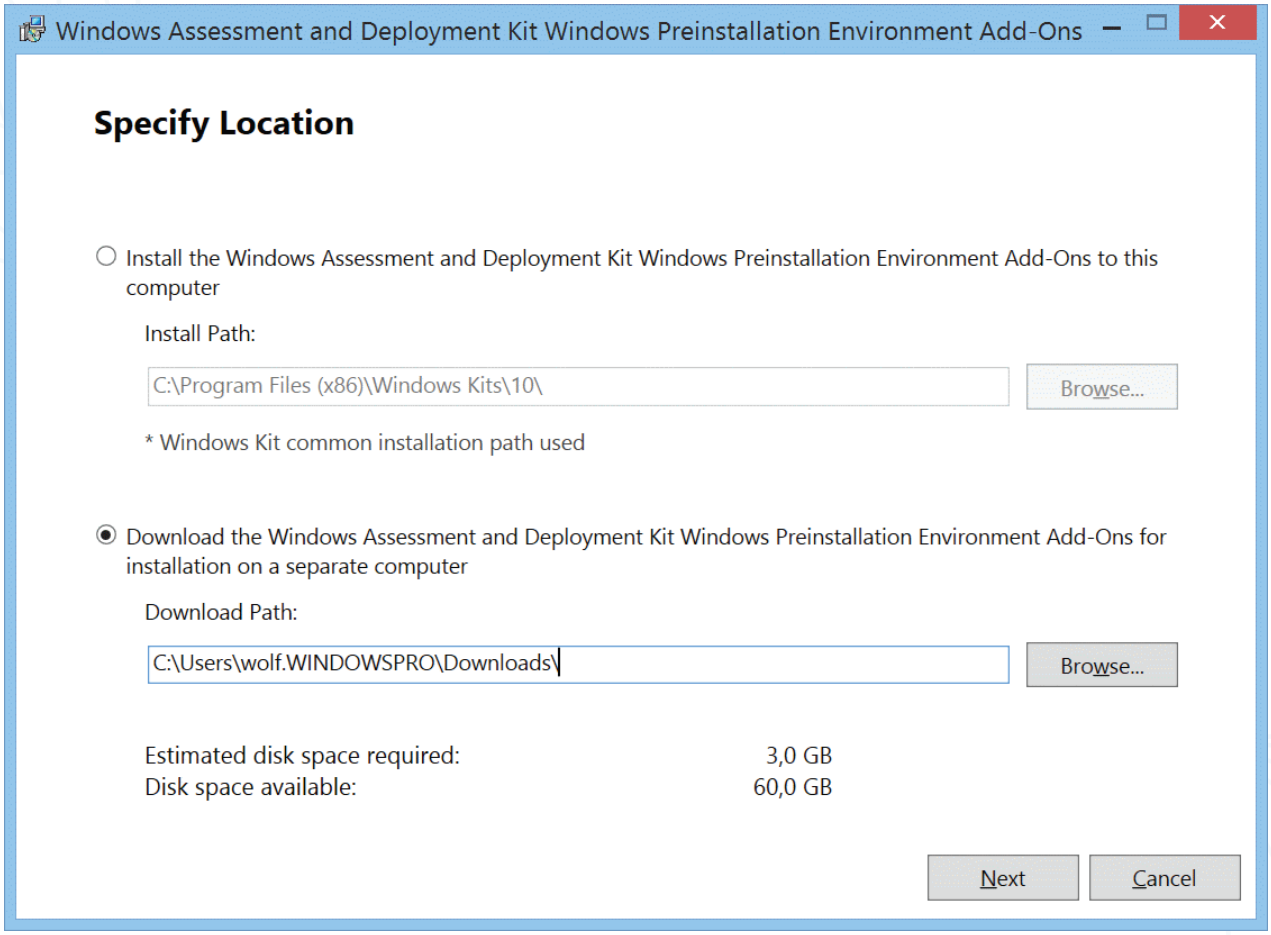
Windows PE war in der Vergangenheit stets ein Bestandteil des WAIK bzw. des Windows ADK, aber seit Windows 10 1809 ist es ein eigener Download. Dieser alleine reicht aber nicht, um eine ISO-Datei oder einen USB-Stick mit WinPE zu erstellen. Vielmehr braucht man weiterhin die Bereitstellungs-Tools aus dem ADK.

WinPE-Paket installieren

Führt man das Setup für Windows PE aus, dann installiert es die Dateien wie schon unter Windows 10 standardmäßig unter

%ProgramFiles(amd64)%\Windows Kits\10\

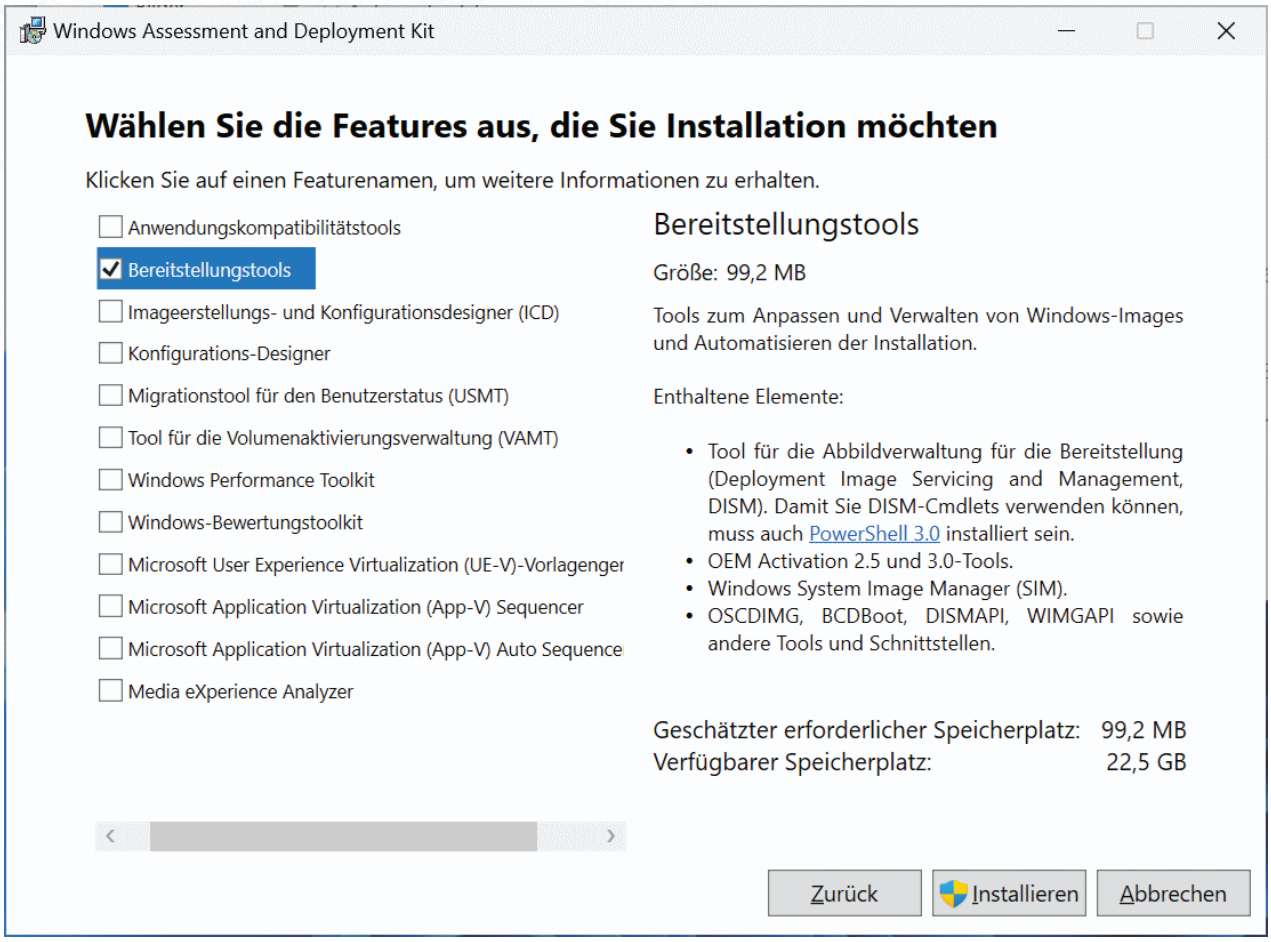
Startet man die Batch-Dateien, die dem Erzeugen eines boot-fähigen WinPE dienen, zu diesem Zeitpunkt, dann brechen diese aufgrund des fehlenden ADK mit diversen Fehlermeldungen ab.



Die Dateien für das Erzeugen eines WinPE-Datenträgers installiert man über ein separates Setup

Bereitstellungstools hinzufügen

Daher startet man im nächsten Schritt das ADK-Setup und wählt dort die Bereitstellungstools zur Installation aus.



Aus dem Windows ADK benötigt man für WinPE nur die Bereitstellungstools

7

Ist dieser Vorgang abgeschlossen, dann öffnet man eine Eingabeaufforderung und führt im Verzeichnis *Deployment Tools* (standardmäßig unter *%ProgramFiles(x86)%\Windows Kits\10\Assessment and Deployment Kit*) die Batch-Datei

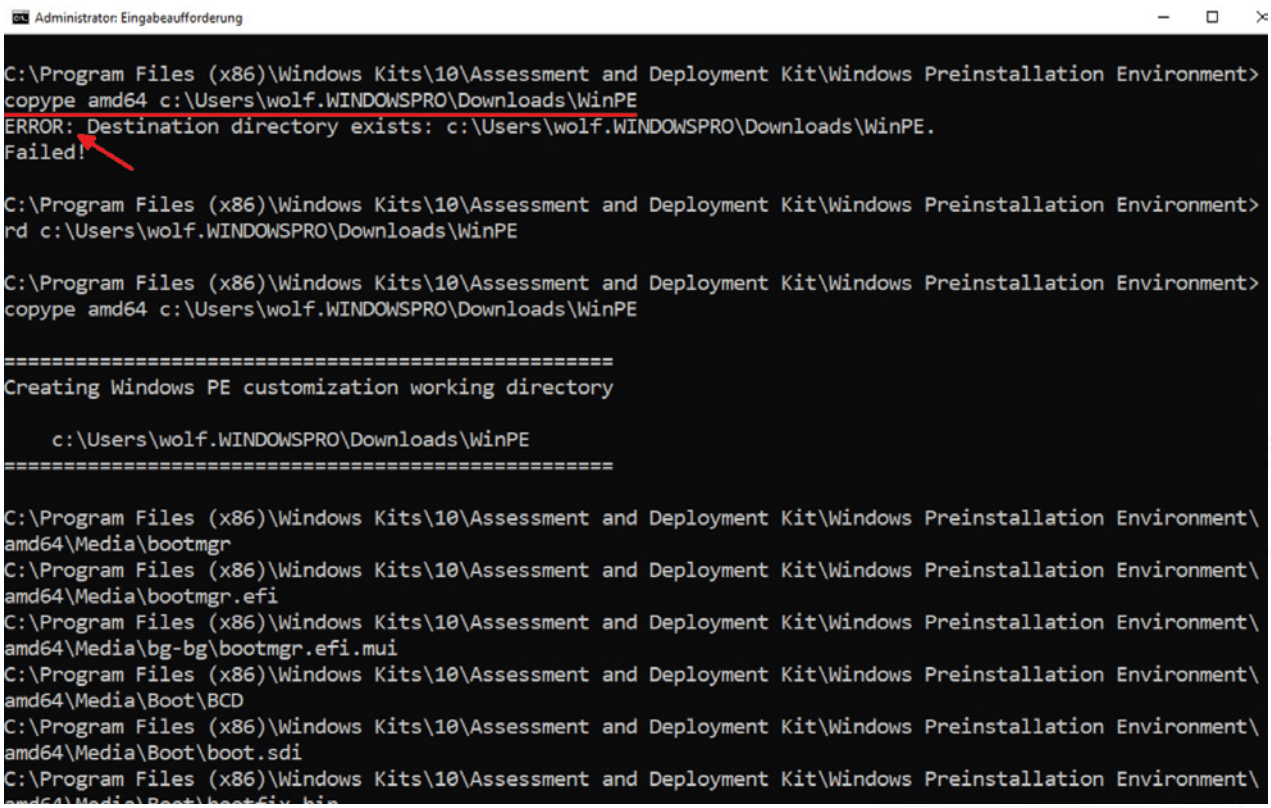
DandISetEnv.bat

aus. Sie setzt die benötigten Umgebungsvariablen.

Nun stellt man die Dateien bereit, die man für die ISO oder den Memory-Stick benötigt. Diesem Zweck dient copype.cmd, ein Aufruf könnte so aussehen:

copype amd64 %USERPROFILE%\WinPE

Der erste Parameter spezifiziert die Architektur, in diesem Fall 64-Bit für Intel bzw. AMD. Zur Auswahl stehen noch *x86*, *arm* und *arm64*. Über den zweiten Parameter gibt man das Zielverzeichnis an. Dieses darf jedoch nicht existieren, sonst bricht der Vorgang ab.



Dateien für das Boot-Medium mit copype.cmd bereitstellen

ISO oder bootfähigen USB-Stick erzeugen

Grundsätzlich sind damit schon die Voraussetzungen geschaffen, ein Boot-Medium mit Windows PE zu erzeugen. Folgender Befehl generiert ein ISO-Abbild:

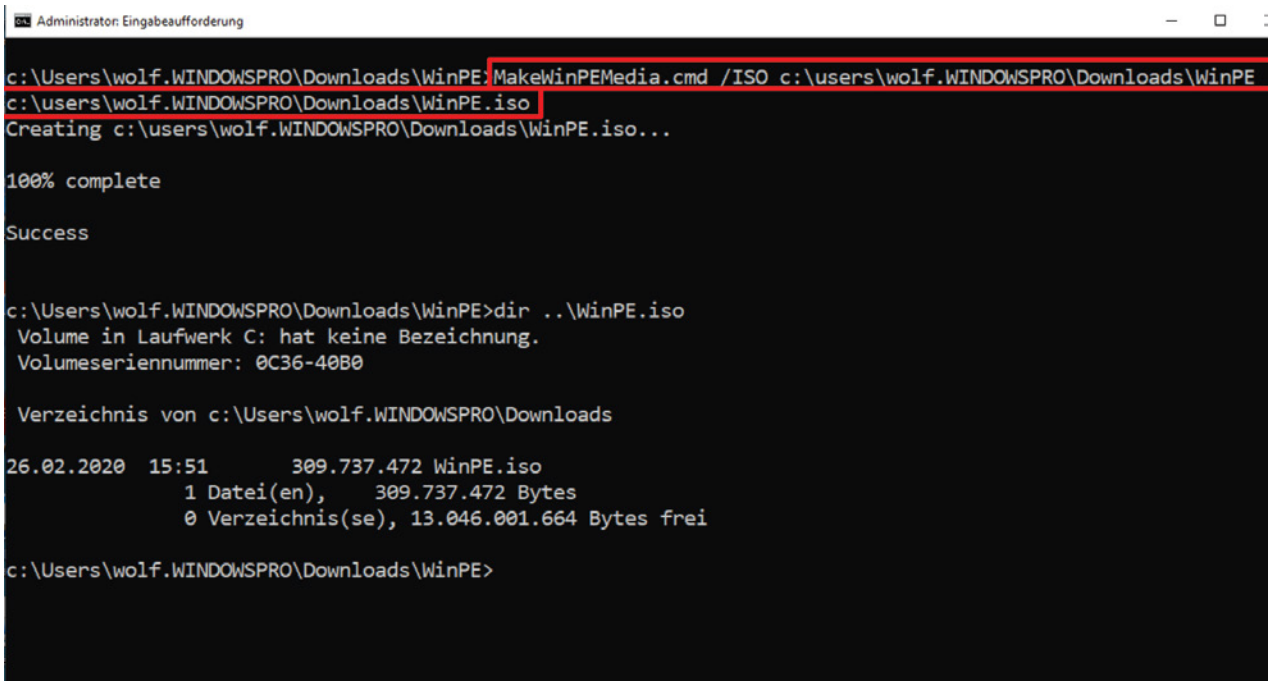
MakeWinPEMedia.cmd /ISO %USERPROFILE%\WinPE .\WinPE.iso

Nach dem Schalter für die ISO folgt das Verzeichnis, das man bei *copype.cmd* als Ziel angegeben hat und zum Schluss steht der Name der ISO-Datei.

Das Erzeugen eines USB-Sticks erfolgt auf die gleiche Weise, nur dass man dafür den Schalter UFD nutzt und als Ziel bloß den Buchstaben des Laufwerks angibt:

MakeWinPEMedia /UFD c:\Users\me\WinPE G:

Wenn man die Warnung vor der Formatierung des Sticks unterdrücken möchte, kann man dem Aufruf den Schalter /F hinzufügen.



Boot-Medium für Windows PE mit MakeWinPEMedia.cmd erzeugen

Nach dem erfolgreichen Abschluss des Vorgangs erhält man ein boot-fähiges, aber nur minimalistisches WinPE. Es enthält nur das englische Sprachpaket, die englische Tastaturbelegung und keinerlei Tools. Daher wird man in der Regel vor dem Generieren des Datenträgers das Image für Windows PE anpassen.

WinPE-Abbild mit DISM bereitstellen

Im Verzeichnisbaum unterhalb des Zielordners von *copype.cmd* findet sich in media\sources die Datei *boot.wim*. Möchte man Windows PE anpassen, dann muss man das WIM mit DISM mounten.
Dies erfolgt auf der Eingabeaufforderung aus dem Verzeichnis, das man bei *copype.cmd* als Ziel angegeben hat, dort existiert das Verzeichnis *mount* bereits:

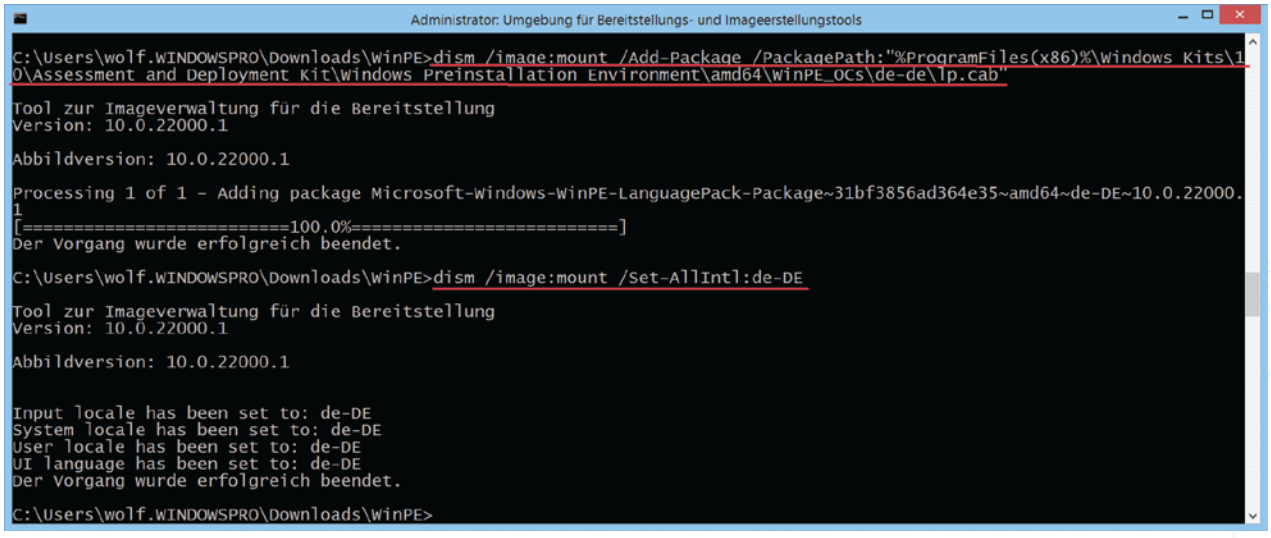
dism /Mount-Wim /MountDir:mount /wimfile:boot.wim /index:1

Sprachpaket hinzufügen

Die Sprachdateien findet man standardmäßig unter *%ProgramFiles(x86)%\Windows Kits\10\Assessment and Deployment Kit\Windows Preinstallation Environment\<Architektur>\WinPE_OCs* in den Verzeichnissen, die nach den Sprachcodes benannt sind.

Das deutsche Sprachpaket fügt man in der x64-Version so hinzu:

dism /image:mount /Add-Package /PackagePath: "%ProgramFiles(amd64)%\Windows Kits\10\Assessment and Deployment Kit\Windows Preinstallation Environment\amd64\WinPE_OCs\de-de\lp.cab"



Deutsches Sprachpaket hinzufügen und das Layout der Tastatur ändern

Mit

dism /image:mount /Set-AllIntl:de-DE

setzt man Sprache der Kommandozeile, das Tastatur-Layout und den Standort auf *Deutschland*.

PowerShell und Tools integrieren

Möchte man zusätzlich PowerShell hinzufügen, dann muss man dafür mehrere Pakete in einer vorgegebenen Reihenfolge kopieren, sie befinden sich im gleichen Verzeichnis wie die Sprachdatei:

Dism /Image:mount /Add-Package
/PackagePath:"%WinPERoot%\amd64\WinPE_OCs\WinPE-WMI.cab"

Dism /Image:mount /Add-Package
/PackagePath:"%WinPERoot%\amd64\WinPE_OCs\de-de\WinPE-WMI_de-de.cab"

Dism /Image:mount /Add-Package
/PackagePath:"%WinPERoot%\amd64\WinPE_OCs\WinPE-NetFx.cab"

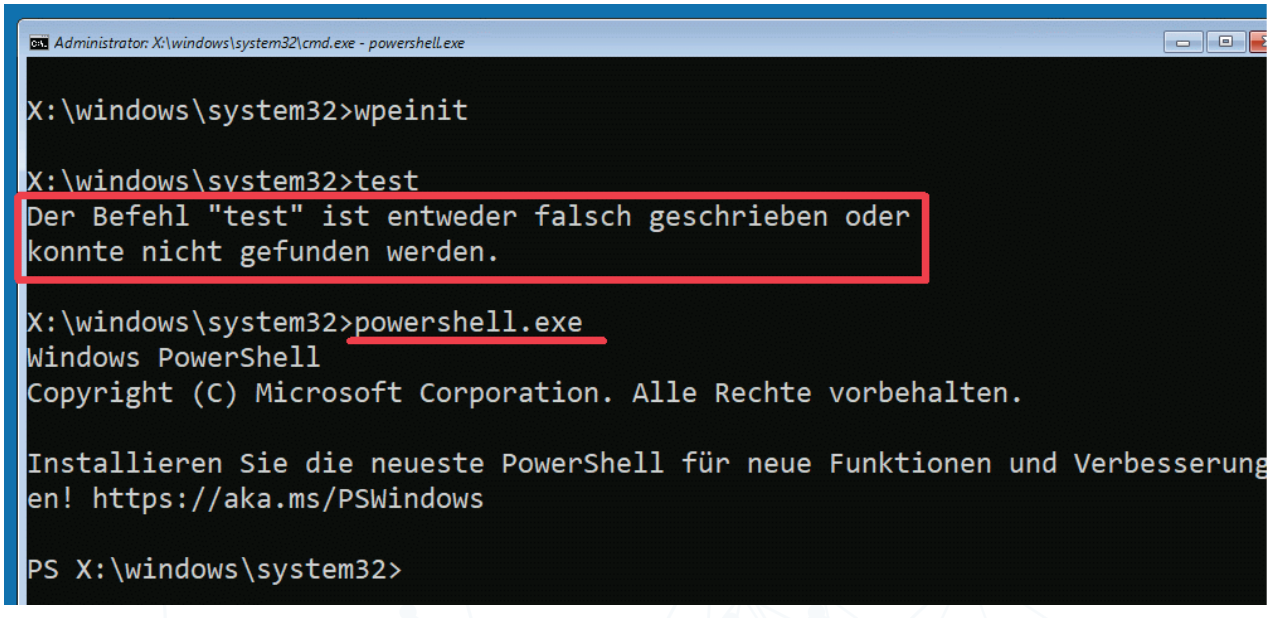
Dism /Image:mount /Add-Package
/PackagePath:"%WinPERoot%\amd64\WinPE_OCs\de-de\WinPE-NetFx_de-de.cab"

Dism /Image:mount /Add-Package
/PackagePath:"%WinPERoot%\amd64\WinPE_OCs\WinPE-PowerShell.cab"

Dism /Image:mount /Add-Package
/PackagePath:"%WinPERoot%\amd64\WinPE_OCs\de-de\WinPE-PowerShell_de-de.cab"

Die Pakete aus dem Verzeichnis de-de dienen der Lokalisierung. Wenn man mit der englischen Version von PowerShell zufrieden ist, benötigt man diese nicht.
Wenn man das Windows-Setup von WinPE starten möchte, dann sollte man noch das Paket *WinPE-SecureStartup* hinzufügen:

Dism /Image:mount /Add-Package
/PackagePath:"%WinPERoot%\amd64\WinPE_OCs\WinPE-SecureStartup.cab"



WinPE mit den deutschen Spracheinstellungen und PowerShell an Bord

Wenn man möchte, kann man noch seine eigenen Tools in das Image kopieren, indem man unter dem mount-Verzeichnis zum gewünschten Ordner wechselt, etwa zum Programmverzeichnis. In Frage kämen beispielsweise die Kommandozeilenprogramme von SysInternals. Allerdings muss man damit rechnen, dass viele davon auf diesem abgespeckten Windows nicht laufen.

Um die Änderungen an der boot.wim zu übernehmen, führt man folgenden Befehl aus:

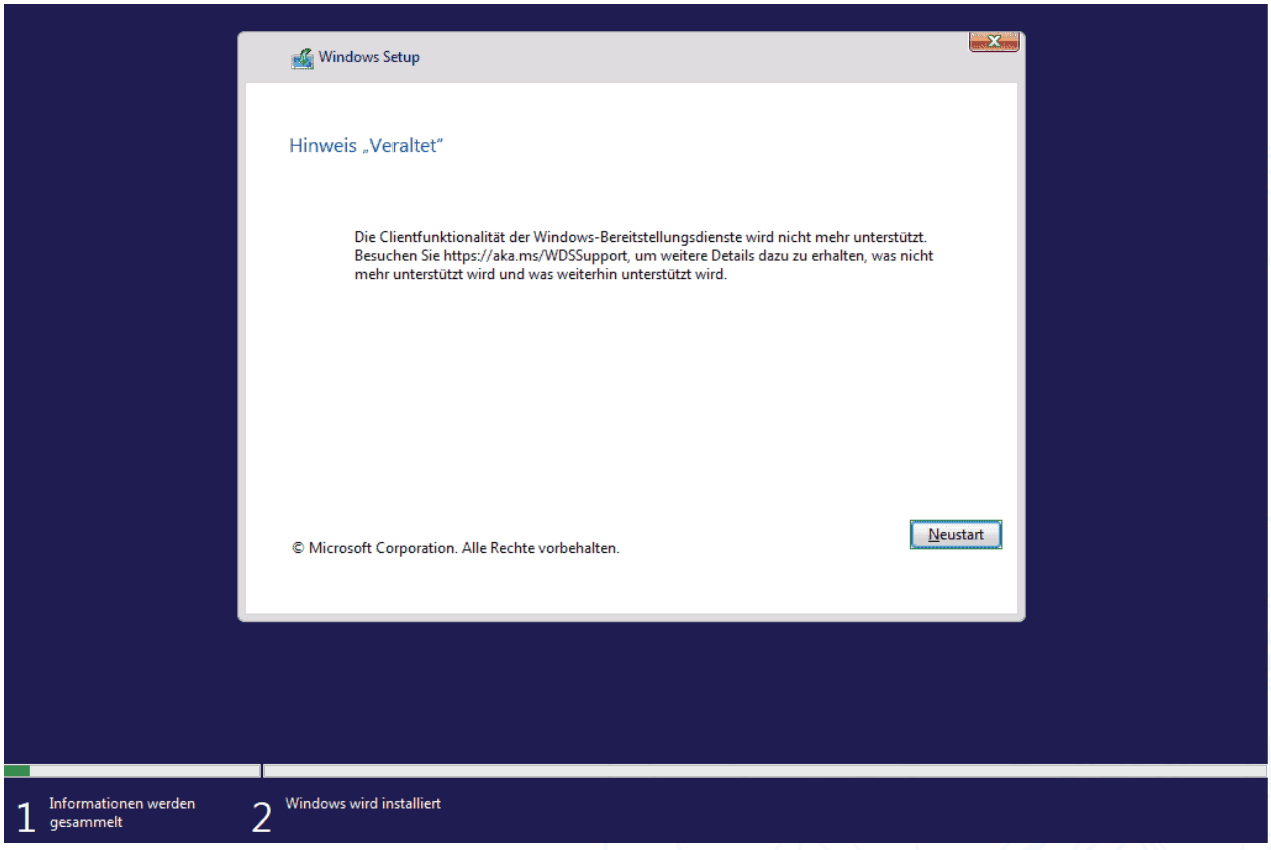
dism /Unmount-Wim /MountDir:mount /commit

Deployment Services (WDS) für WinPE bereitstellen

Die Bereitstellungsdienste (WDS) von Windows Server haben sich seit Jahren als einfach nutzbare Technik für das OS-Deployment etabliert. Die WDS enthalten nicht nur einen PXE-Server, um Clients über das Netzwerk zu booten, sondern lassen sich optional mit dem AD integrieren, so dass man dort schon vorab Computerkonten anlegen kann. Außerdem bieten sie einen automatischen Workflow, der nach dem Booten eines Rechners von einem Image automatisch das Windows-Setup startet.

Für ein Standard-Setup reichten die WDS bisher jedoch alleine aus, wobei man dieses natürlich ebenfalls über eine Antwortdatei automatisieren kann. Diese Option stuft Microsoft für Windows 11 als veraltet ein. Wenn man die boot.wim aus den Installationsmedien auf den WDS-Server hochlädt, dann startet das Setup nicht mehr, vielmehr erhält man den Hinweis:

"Die Client-Funktionalität der Windows-Bereitstellungsdienste wird nicht mehr unterstützt"

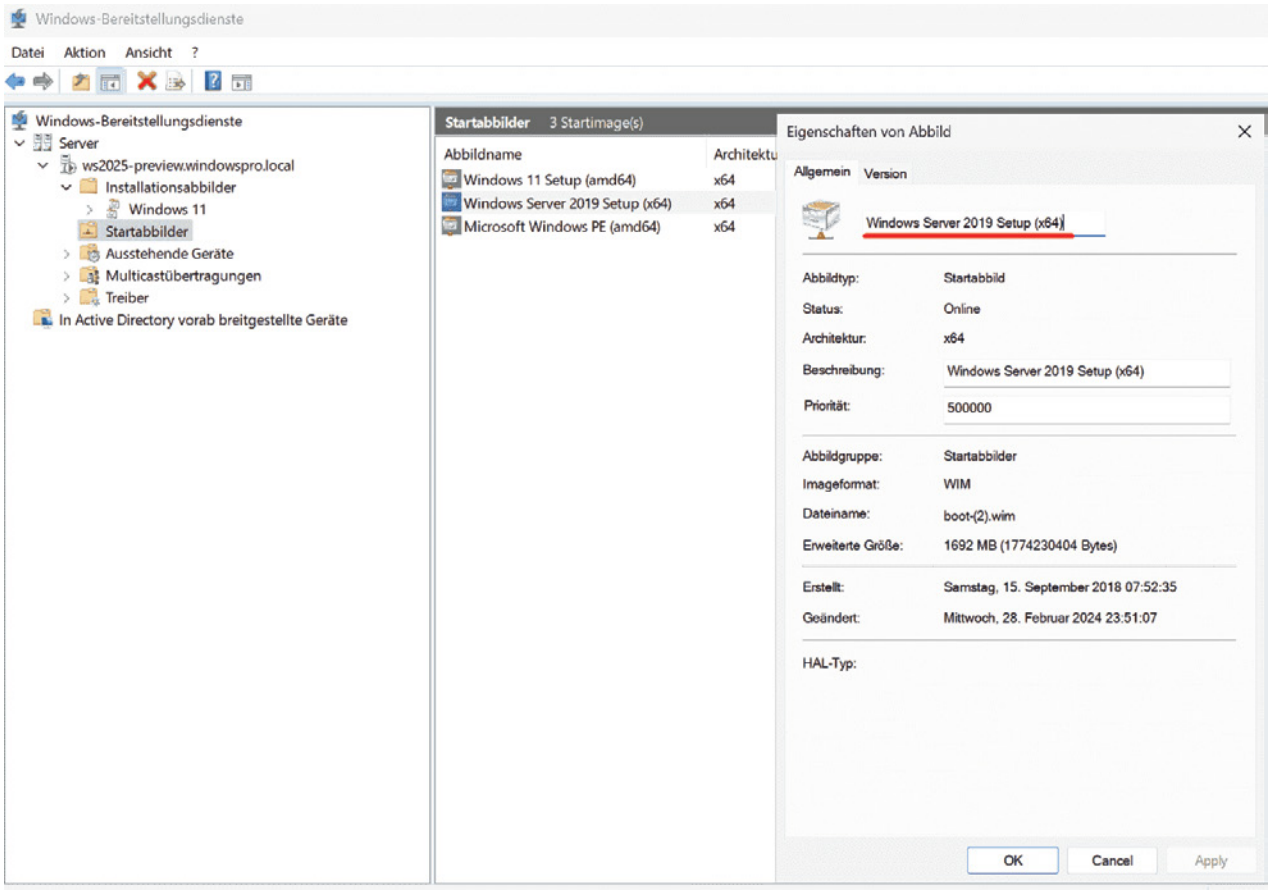


Nach dem Start eines PCs von der Windows 11 boot.wim läuft das Windows-Setup nicht mehr

Auf ältere boot.wim ausweichen

Laut Dokumentation betrifft diese Einschränkung nur die boot.wim aus der Windows-11-ISO, nicht aber WinPE-Images. Außerdem sollte das Setup dieser Matrix zufolge auch dann nicht laufen, wenn man eine ältere boot.wim verwendet.

In der Praxis kann man derzeit jedoch noch Rechner vom Boot-Image für Windows 10 starten, und anschließend läuft das Setup für Windows 11 klaglos durch. Dies klappt sogar dann, wenn man die WDS auf Windows Server 2025 verwendet.



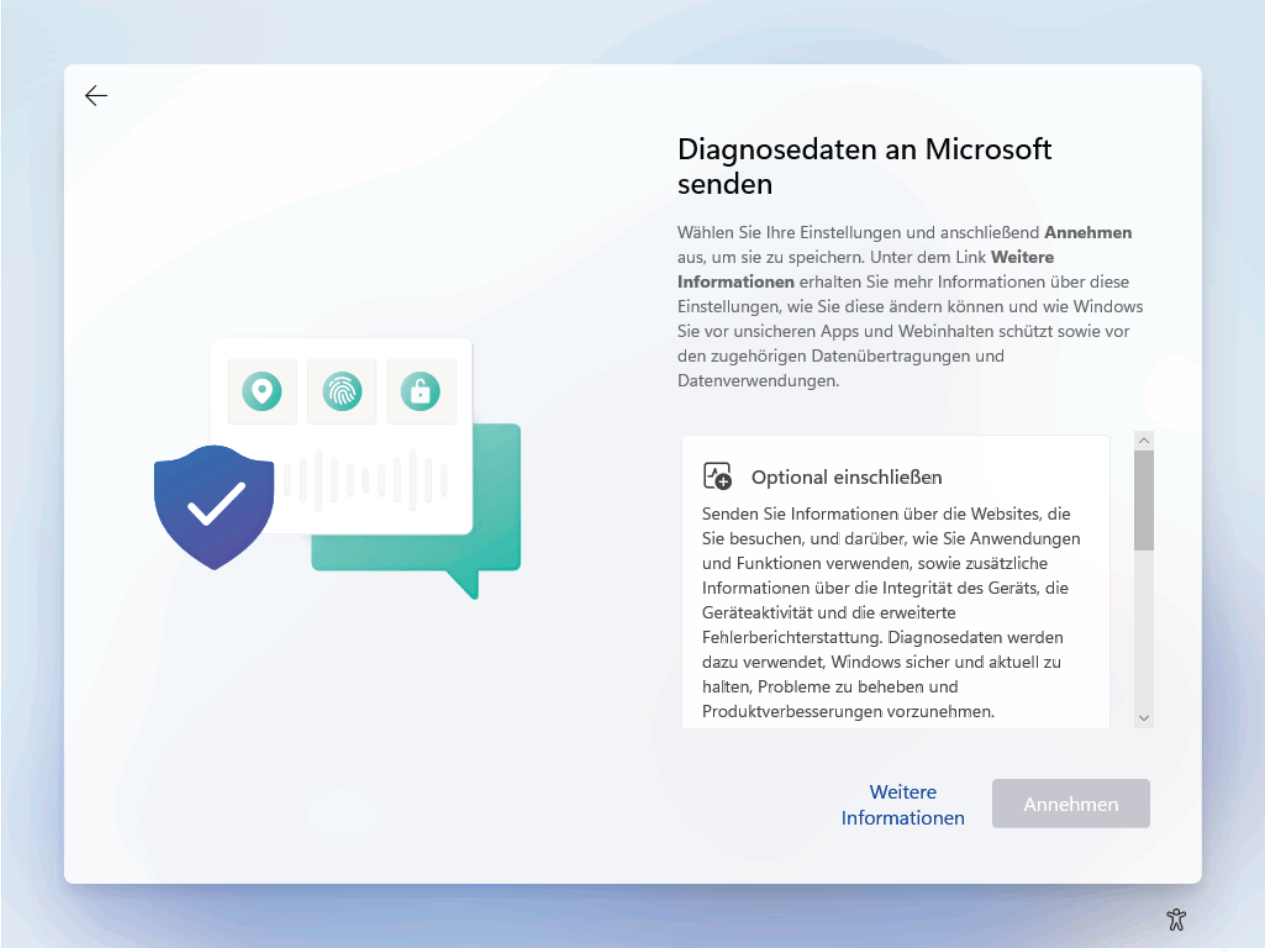
In diesem Beispiel starten die Clients für die Windows-11-Installation von der boot.wim aus Server 2019

Wenn Microsoft irgendwann dieses Schlupfloch schließt, besteht weiterhin die Möglichkeit, ein angepasstes WinPE und einem Setup-Script als Boot-Image zu verwenden, so wie es etwa OSDCloud macht. Dies ist nach wie vor eine interessante Alternative zum Booten von einem externen Datenträger.

OOBE-Dialoge mit unattend.xml beantworten

Unabhängig davon, ob man Windows manuell, automatisiert mit Hilfe einer Antwortdatei oder durch Anwenden eines mit Sysprep generalisierten Custom-Images installiert, als Erstes sehen die Benutzer immer die OOBE-Dialoge, wenn sie das neue OS zum ersten Mal booten.

Diese fragen unter anderem die gewünschten Einstellungen für die Privatsphäre ab. Dazu zählen der Umgang mit den Telemetriedaten, die Verwendung der Werbe-ID, des Standorts oder Informationen zum Auffinden des Geräts im Falle eines Verlusts.

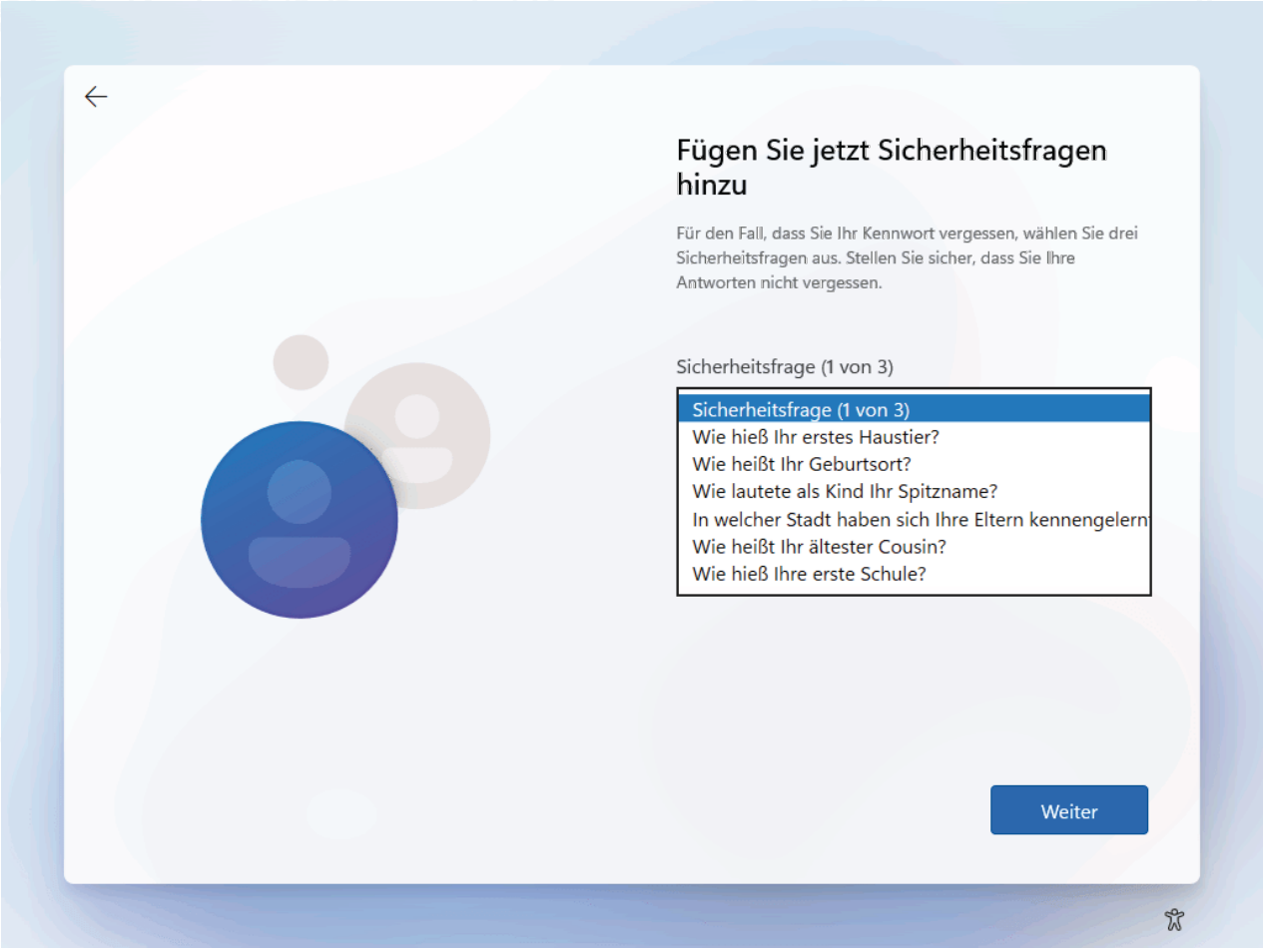


Über den Transfer von Diagnosedaten an Microsoft sollten nicht die Endbenutzer bestimmen

In verwalteten Umgebungen ist es fast immer unerwünscht, dass die Benutzer selbst Konten anlegen, mit denen sie sich anschließend anmelden. Zwar macht Microsoft das Erstellen von lokalen Accounts zum Versteckspiel, aber beim erfolgreichen Absolvieren dieses Schritts würden die User über einen lokalen Admin verfügen.

Das Anmelden mit einem Microsoft-Konto oder das Anlegen eines lokalen Admins durch den User ist auf Firmen-PCs meist unerwünscht

Damit sich die Benutzer gleich mit ihrem Domänen-Konto anmelden können, muss man den Rechner bereits in einer früheren Setup-Phase der Domain anschließen. Im OOBE-Abschnitt ist das nicht mehr vorgesehen.



Das Anlegen eines lokalen Kontos ist mit dem nervigen Beantworten von drei Sicherheitsfragen verbunden

Die Automatisierung der OOBE-Phase verhindert nicht nur ungünstige Systemeinstellungen, sondern erspart den Benutzern eine zeitraubende Aufgabe. Dies kommt auch Admins oder fortgeschrittenen Anwendern entgegen, wenn sie etwa regelmäßig neue virtuelle Maschinen mit Windows 11 anlegen.

Antwortdatei für OOBE erstellen

Um eine Antwortdatei zu erzeugen, benötigt man den Windows System Image Manager (Windows SIM). Das Tool ist Bestandteil des Assessment and Deployment Toolkit (ADK). Damit es die benötigten Einstellungen anzeigt, öffnet man ein Installationsabbild, typischerweise die install.wim von Microsofts ISO-Datei.

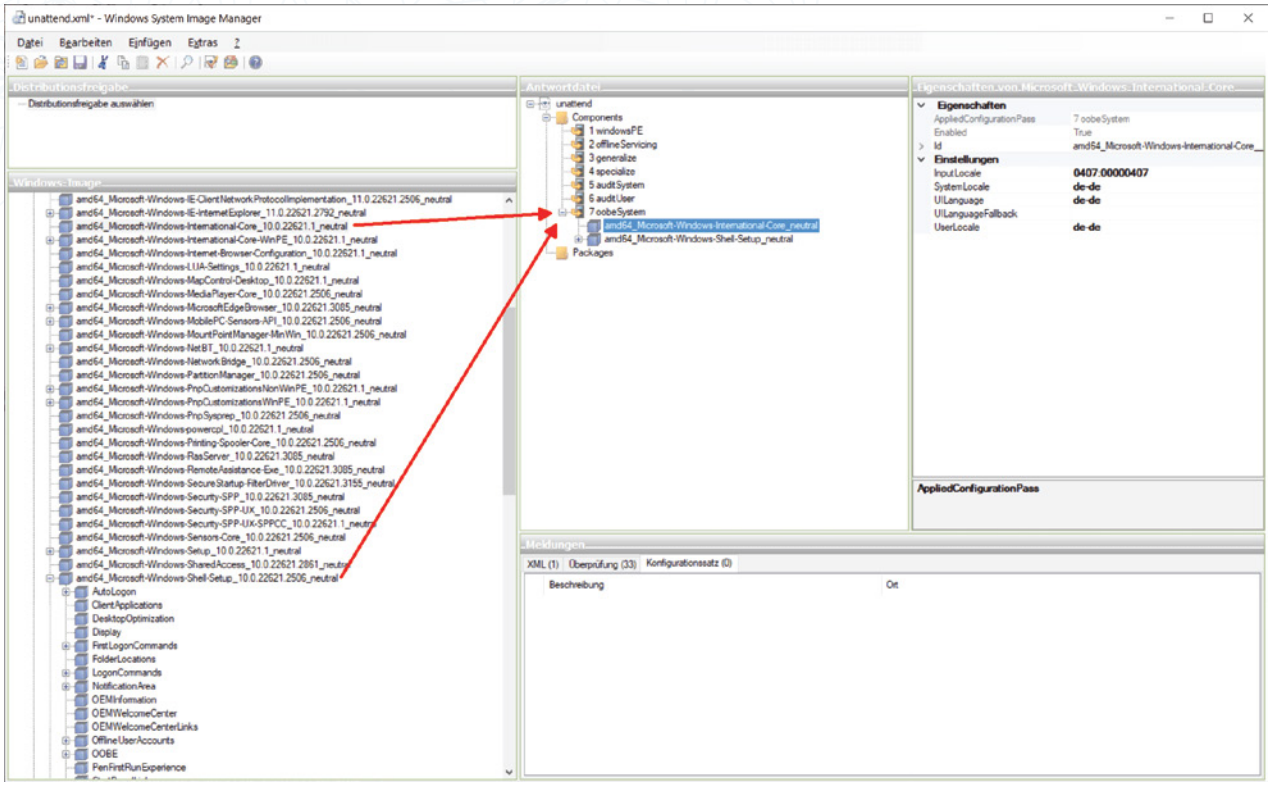
Wenn man das Tool zum ersten Mal ausführt, dann fragt es nach, ob es eine Katalogdatei erstellen soll. Dies sollte man bejahen, wobei man für diesen Vorgang administrative Rechte und etwas Geduld benötigt.

Anschließend zieht man mit der Maus die Einträge

amd64_Microsoft-Windows-International-Core_10.0.<Build-Nummer>_neutral

amd64_Microsoft-Windows-Shell-Setup_10.0.<Build-Nummer>_neutral

vom Fenster links unten auf 7 *oobeSystem* im mittleren Fenster (Vorsicht, es gibt diese Container auch mit dem Präfix *wow64*!).

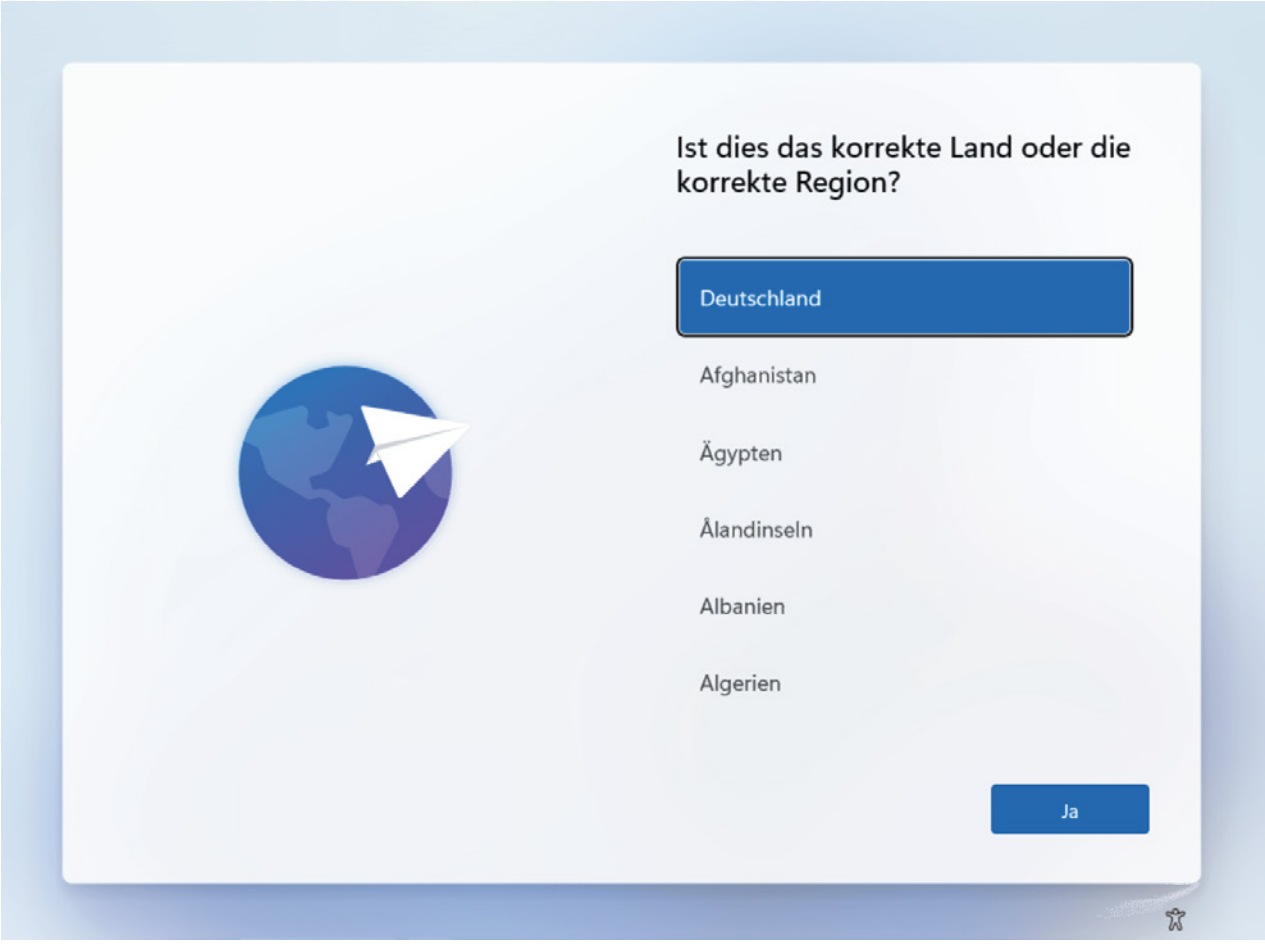


Komponenten mit den benötigten Einstellungen für OOBE übernehmen

Nun erscheint an dieser Stelle die Baumansicht mit allen darin enthaltenen Einstellungen.

Einstellungen für Sprache und Region

Während der OOBE-Phase fragt das Setup die gewünschte Landessprache, das Tastatur-Layout oder die Region ab, um etwa das Zeitformat festzulegen. Diese Einstellungen lassen sich über die Komponente *amd64_Microsoft-Windows-International-Core__neutral* automatisch konfigurieren.

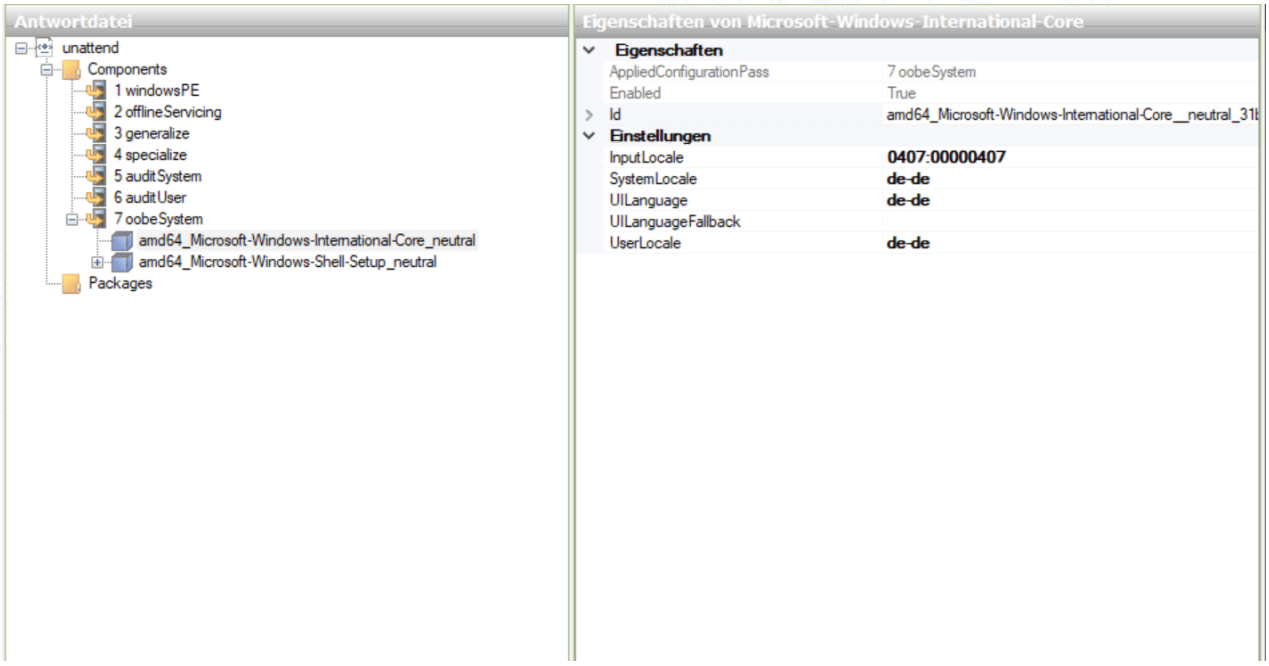


Die Abfragen zu Region und dazugehörigen Einstellungen lassen sich automatisch beantworten

Dort füllt man folgende Felder aus:

- InputLocale (Sprache für Eingabegeräte, Tastatur-Layout)
- SystemLocale (Standardsprache für Nicht-Unicode-Programme)
- UserLocale (bestimmt die Formatierung von Zeit, Datum, Währungen und Zahlen pro User)
- UILanguage (Sprache der grafischen Oberfläche)

Alle Einstellungen kann man nach RFC 3066 spezifizieren, also etwa de-DE für Deutschland oder en-US für die USA. *InputLocale* akzeptiert auch hexadezimale Werte, eine Liste aller Codes findet sich auf Microsoft Docs. Hier kann man auch mehrere Werte, getrennt durch Semikolon, eingeben.



Konfiguration der regionalen Einstellungen im Windows System Image Manager

Dialoge für EULA und Benutzerkonten

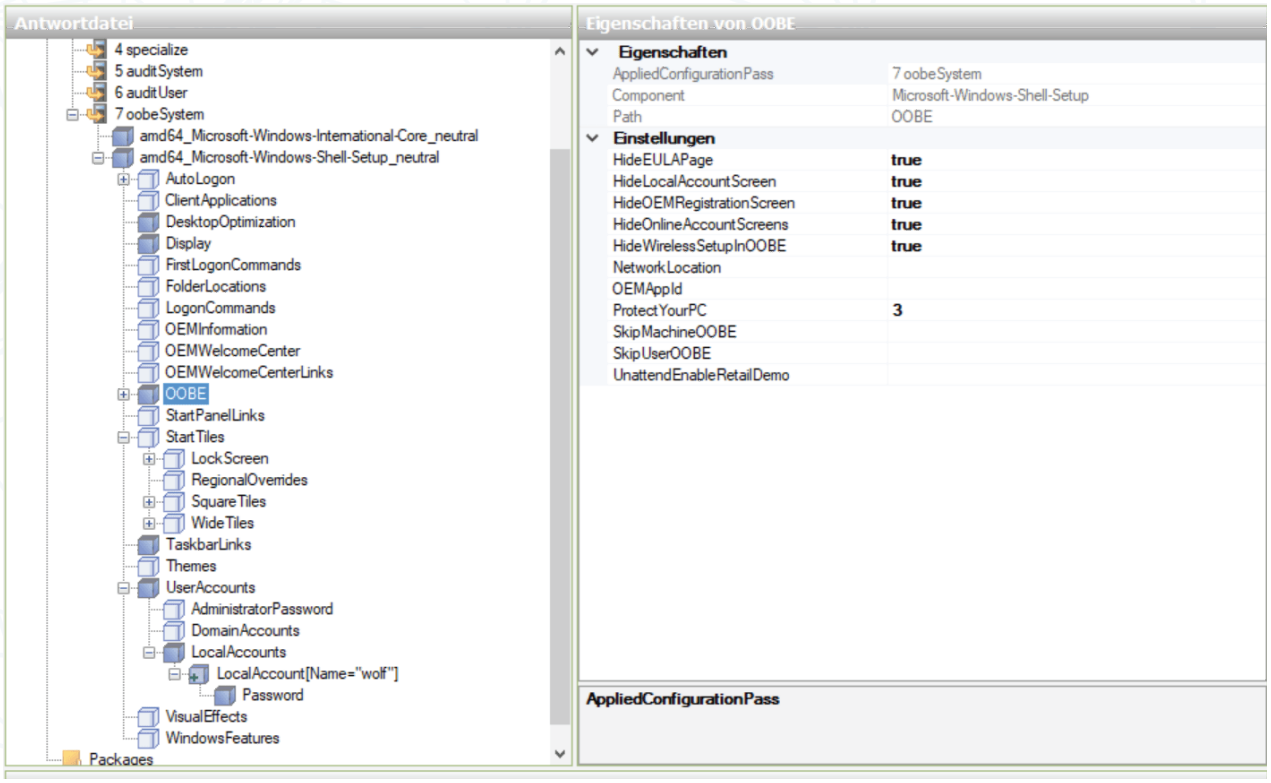
Dafür klappt man die Komponente *amd64_Microsoft-Windows-Shell-Setup__neutral* auf und wechselt zum Abschnitt OOBE, wo man die diversen Dialoge während der Installation ausblenden kann. Diesem Zweck dienen:

- HideEULAPage
- HideOEMRegistrationScreen
- HideWirelessSetupInOOBE
- HideOnlineAccountScreens
- HideLocalAccountScreen

Sie alle setzt man auf den Wert *True*. Die erste Option überspringt die Bestätigung der Lizenzbedingungen, die beiden letzten die Dialoge zum Erstellen eines Kontos (online und lokal).

Einstellungen für die Privatsphäre

Die Konfiguration von Einstellungen zur Privatsphäre, darunter welche Daten an Microsoft gesendet werden oder ob Anwendungen auf den Standort des Benutzers zugreifen dürfen, kann man überspringen, indem man ProtectYourPC den Wert 3 zuweist.

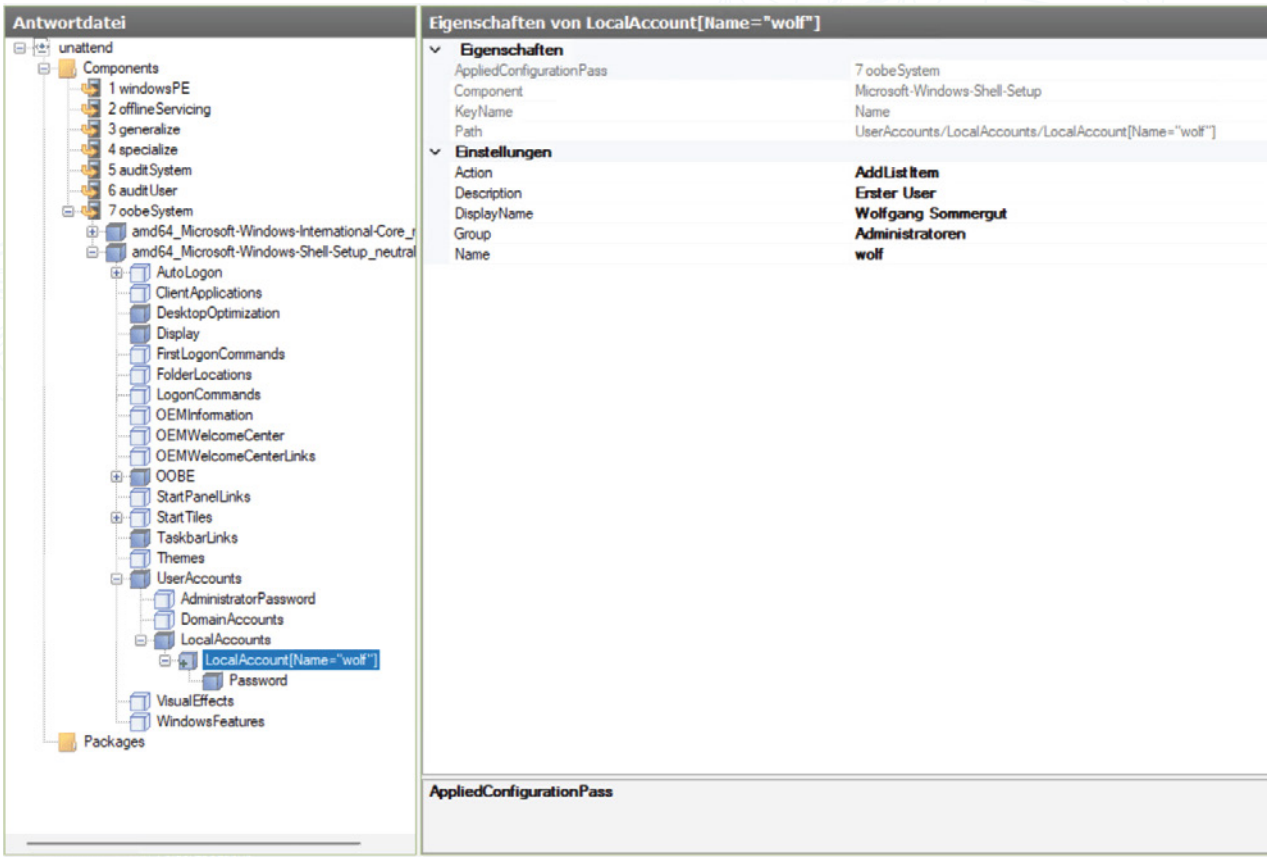


Dialoge für EULA, zum Anlegen eines Kontos und zu den Einstellungen für die Privatsphäre überspringen

Möchte man diese Einstellungen künftig zentral verwalten, dann kann man das über die Gruppenrichtlinien tun.

Lokales Konto anlegen

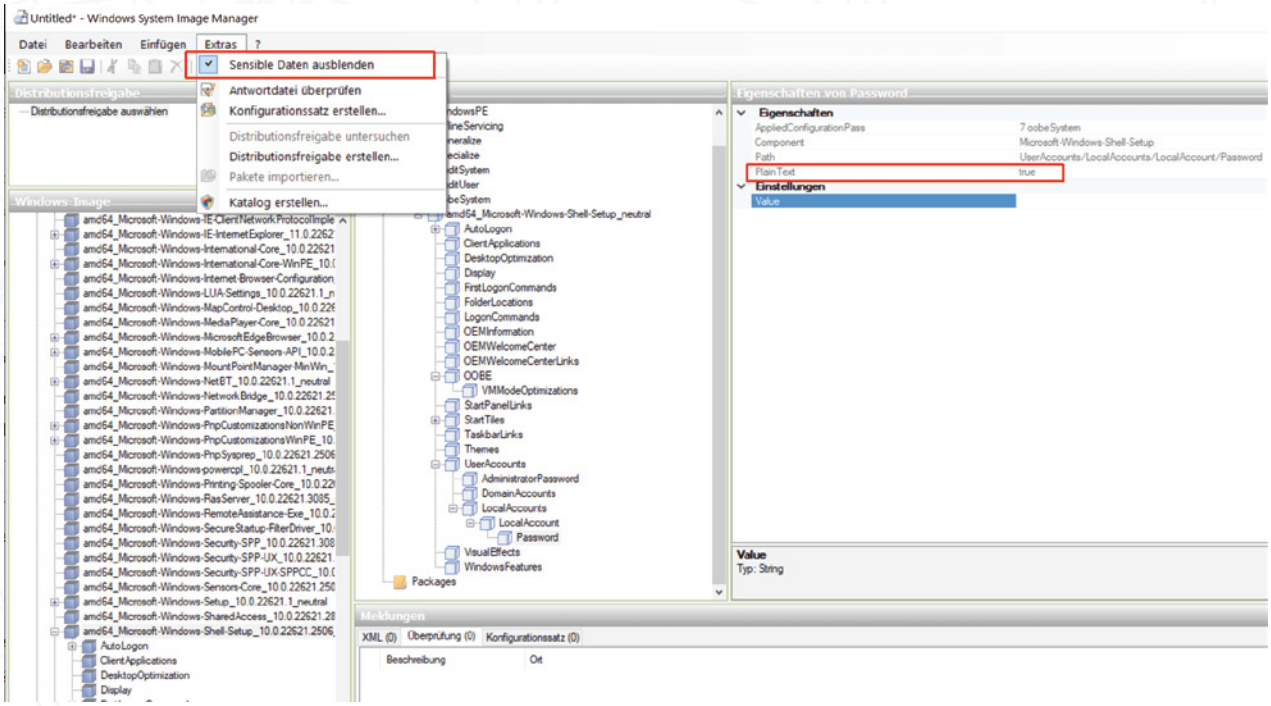
Wenn man die Dialoge zum Anlegen von Benutzerkonten ausgeblendet hat, wird man wahrscheinlich einen Account über die Antwortdatei anlegen. Andernfalls existieren sonst auf dem Rechner nur der deaktivierte Administrator und einige Systemkonten.



Neues lokales Konto beim Setup anlegen

Dies kann man unter *UserAccounts => LocalAccounts* erledigen. Dabei besteht auch die Möglichkeit, gleich das Passwort zu speichern. Wenn man ein administratives Konto automatisch mit einer Antwortdatei anlegt, dann sollte man dessen Kennwort in AD-Umgebungen anschließend mit LAPS zentral verwalten.

Bei einer neuen Antwortdatei steht der Wert für *PlainText* auf *True*, so dass man das Kennwort einfach eintippen kann. Wenn man nicht möchte, dass es dann als Klartext in der unattend.xml gespeichert wird, dann muss man im Menü *Extras* den Haken bei *Sensible Daten ausblenden* setzen.

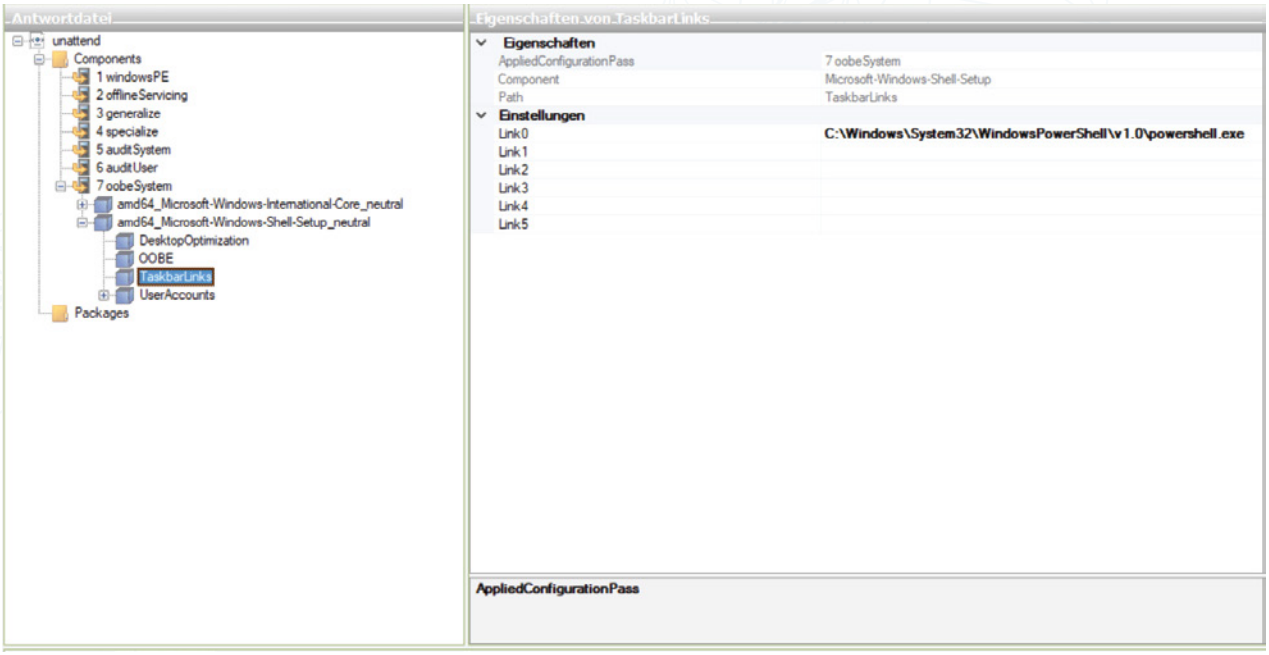


Passwort in der Antwortdatei verschlüsselt speichern

Möchte man die Datei später wieder bearbeiten, dann enthält sie das verschlüsselte Kennwort und *PlainText* hat den Wert *False*. Um das Passwort ändern zu können, muss man daher erst in der XML-Datei den Inhalt des Elements <PlainText> auf *true* setzen.

Weitere Einstellungen

Unter *Microsoft-Windows-Shell-Setup* finden sich noch weitere Optionen, die für die OOBE-Automatisierung interessant sein können. Die Benutzer müssen diese Einstellungen während des Setups jedoch nicht konfigurieren, aber man kann auf diesem Weg das System noch weiter an die jeweiligen Anforderungen anpassen.



Programme über die Antwortdatei an die Taskleiste anheften

So kann man mit der Option *TaskbarLinks* bis zu fünf Programme an die Taskleiste anheften. Außerdem lässt sich hier ein Benutzerkonto für die automatische Anmeldung hinterlegen. Unter *DesktopOptimization* kann man dafür sorgen, dass vorinstallierte Store Apps nicht auf der Taskleiste erscheinen.

Eine Reihe von Einstellungen ist indes veraltet. Dies betrifft die Konfiguration des Startmenüs und *VisualEffects*. Außerdem sollte man *FolderLocations* nur in Testumgebungen verwenden, weil das Verschieben der Profile auf ein anderes Volume zu Problemen führt.

Antwortdatei an Image zuweisen

Microsoft bietet mehrere Möglichkeiten, für die unbeaufsichtigte Installation einem Image eine Antwortdatei zuzuteilen. Unabhängig davon, für welche Variante man sich entscheidet, sollte man die Datei unter dem Namen *unattend.xml* speichern.

In den meisten Fällen wird man sie aber in das Systemabbild integrieren, wobei das Setup auch hier mehrere Verzeichnisse zur Auswahl lässt. In Frage kommen hier etwa %SystemRoot%\system32\panther oder das Wurzelverzeichnis des Systemlaufwerks. Alternativ kann man diese auch auf einem USB-Stick bereitstellen.

Nutzt man als Installationsmedium eine ISO-Datei, wie dies beim Einrichten von Windows als Gast-OS meistens der Fall ist, dann muss man deren Inhalt in eines der vorgesehenen Verzeichnisse kopieren, die Antwortdatei hinzufügen und danach die gesamte Struktur wieder in eine ISO packen.

Muster für unattend.xml

Wenn man nur die hier besprochenen Einstellungen per Antwortdatei konfigurieren möchte, dann muss man sich dafür nicht durch Windows SIM arbeiten. Vielmehr kann man einfach die folgende unattend.xml an die eigene Umgebung anpassen.

Das Kennwort für den lokalen Account lautet hier "P@ssword". Wenn man es in der Antwortdatei ändern möchte, dann sollte man so vorgehen wie oben beschrieben,

```
<?xml version="1.0" encoding="utf-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
    <settings pass="oobeSystem">
        <component name="Microsoft-Windows-Shell-Setup" processorArchitecture="amd64"
publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS"
xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <DesktopOptimization>
                <ShowWindowsStoreAppsOnTaskbar>false</ShowWindowsStoreAppsOnTaskbar>
            </DesktopOptimization>
            <OOBE>
                <HideEULAPage>true</HideEULAPage>
                <HideLocalAccountScreen>true</HideLocalAccountScreen>
                <HideOEMRegistrationScreen>true</HideOEMRegistrationScreen>
                <HideOnlineAccountScreens>true</HideOnlineAccountScreens>
                <HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>
                <ProtectYourPC>3</ProtectYourPC>
            </OOBE>
            <TaskbarLinks>
                <Link0>C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe</Link0>
            </TaskbarLinks>
            <UserAccounts>
                <LocalAccounts>
                    <LocalAccount wcm:action="add">
                        <Password>
                            <Value>UABAAHMAcwB3AG8AcgBkAFAAYQBzAHMAdwBvAHIAZAA=</Value>
                            <PlainText>false</PlainText>
                        </Password>
```

```
                <Description>Erster User</Description>
                <DisplayName>Max Mustermann</DisplayName>
                <Group>Administratoren</Group>
                <Name>max</Name>
              </LocalAccount>
            </LocalAccounts>
          </UserAccounts>
        </component>
        <component name="Microsoft-Windows-International-Core" processorArchitecture="amd64"
publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS"
xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <InputLocale>0407:00000407</InputLocale>
          <UILanguage>de-de</UILanguage>
          <UserLocale>de-de</UserLocale>
          <SystemLocale>de-de</SystemLocale>
        </component>
    </settings>
    <cpi:offlineImage cpi:source="wim://c:/users/public/downloads/iso/sources/install.wim#Windows 11
Enterprise" xmlns:cpi="urn:schemas-microsoft-com:cpi" />
</unattend>
```

Image erfassen und warten

Sowohl die Installation via Setup als auch das Aufspielen eines Images mit DISM benötigen ein Systemabbild in Form eines WIM-Archivs.

Im ersten Fall kann man es bei der install.wim belassen, so wie sie von Microsoft ausgeliefert wird, oder ein individuelles Image verwenden. Erstellt man eine Musterinstallation, um diese als WIM zu erfassen, dann tut man dies natürlich, um das Image an die eigenen Erfordernisse anzupassen.

Auch hier erfordern beide Methoden gleichermaßen eine regelmäßige Wartung des Images. Dazu gehört das Einspielen von Updates und Treibern oder das Aktualisieren von Apps.

Custom Image von Windows 11 generalisieren und erfassen

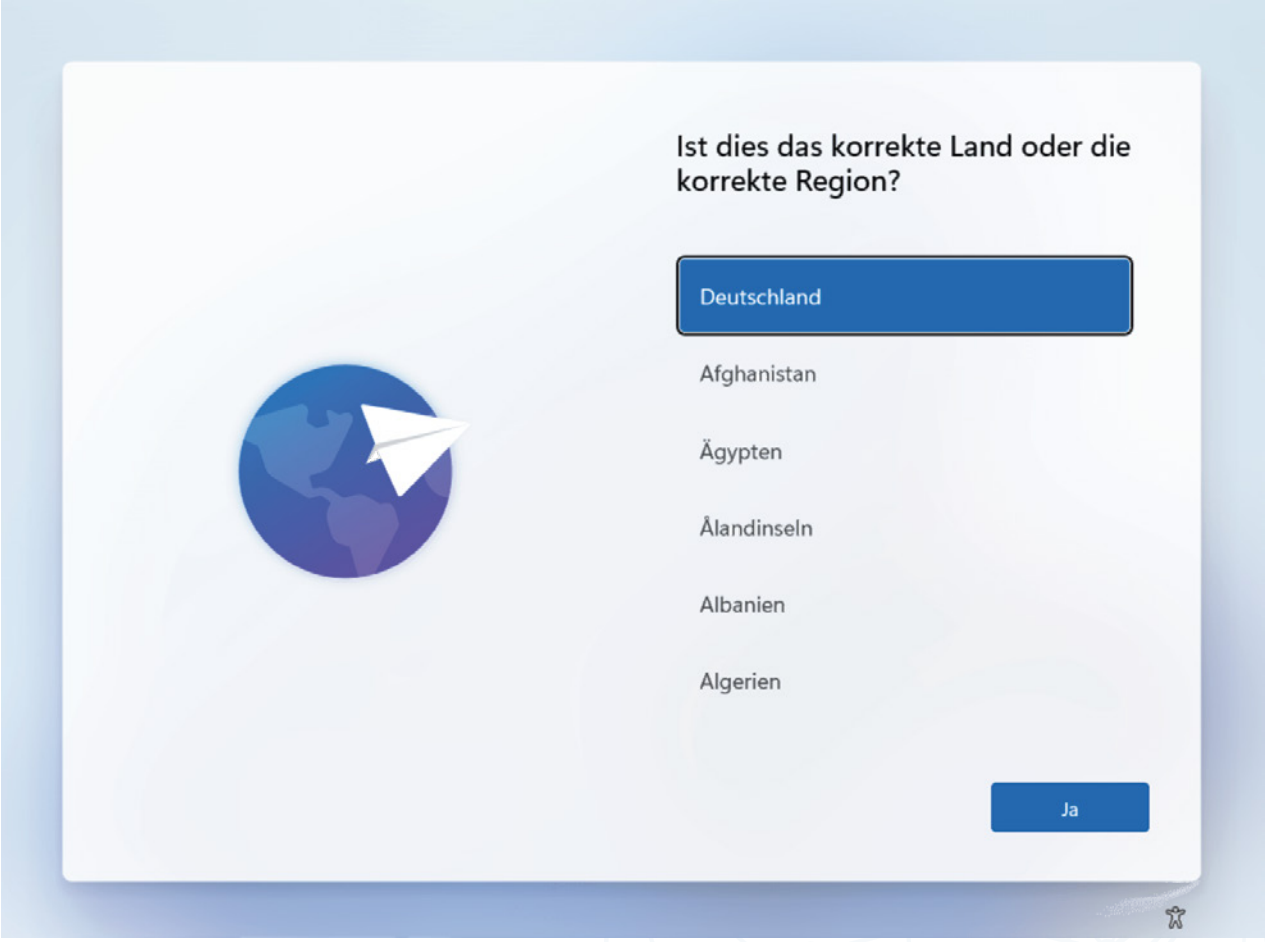
Microsoft sieht verschiedene Wege zu einem angepassten Installationsabbild für Windows 11 vor.

Möchte man nur Features, Updates und Treiber hinzufügen oder Apps entfernen, dann kann man dies auch offline tun. Will man das OS darüber hinaus anpassen, dann muss man es installieren.

Für diese Aufgabe nimmt man heute keine physischen PCs mehr, sondern virtuelle Maschinen. Ein Vorteil beim Einsatz von Hyper-V, VMware Workstation oder VirtualBox besteht darin, dass man vor dem Ausführen von Sysprep einen Snapshot der Referenzinstallation erstellen kann. Auf diese Weise lässt sich das Image laufend aktualisieren und immer wieder verwenden.

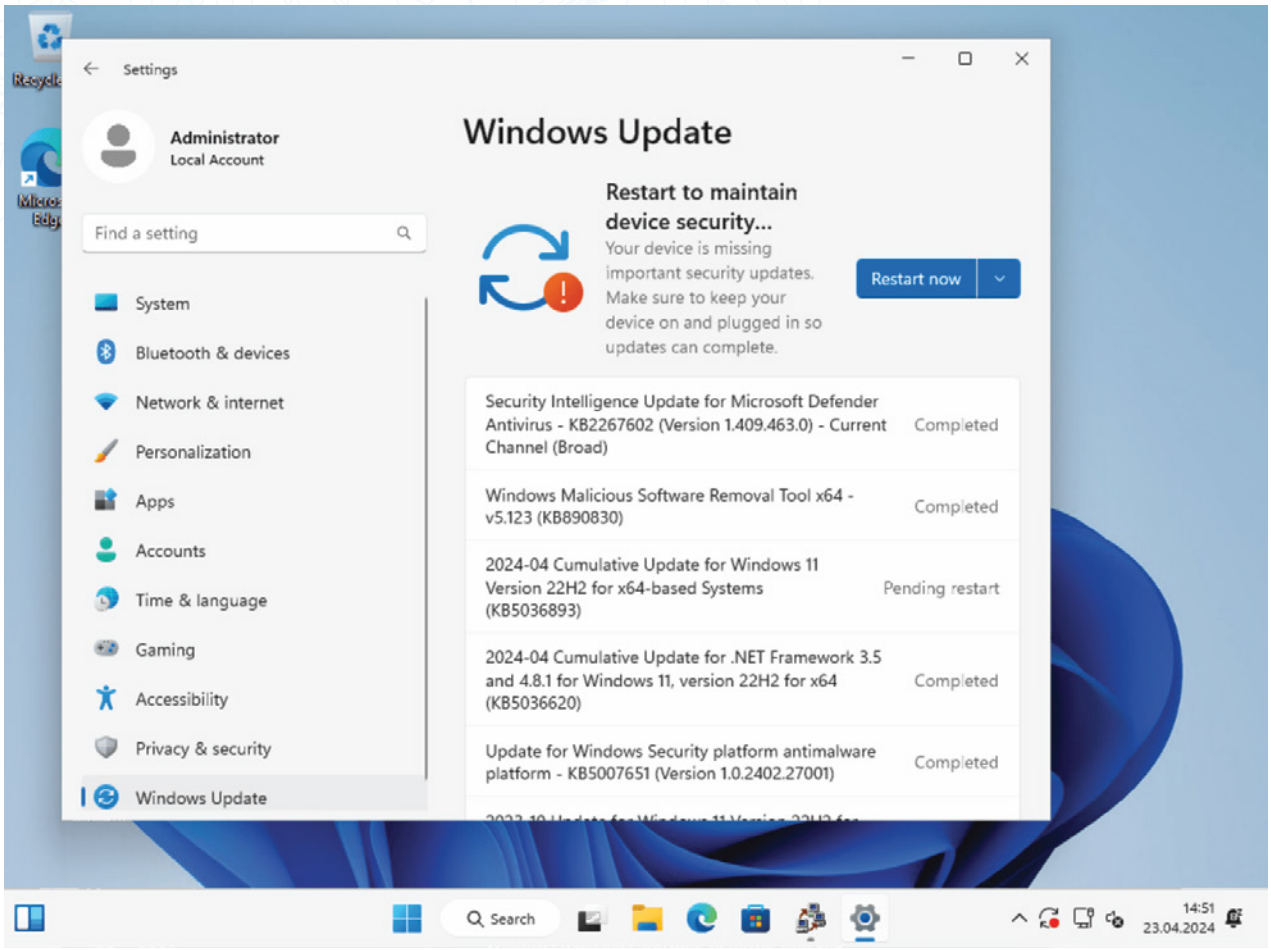
Windows 11 im Audit Mode anpassen

Nach der Installation von Windows 11 bootet es schließlich in die OOBE-Phase. Für die geplante Anpassung des Images benötigt man aber den Audit Mode. Diesen aktiviert man am einfachsten, indem man beim ersten OOBE-Bildschirm zur Auswahl der Ländereinstellungen die Tastenkombination STRG + Umschalten + F3 betätigt.



Beim ersten OOBE-Dialog kann man den Reboot von Windows 11 in den Audit Mode veranlassen

Die virtuelle Maschine startet nun neu und meldet das eingebaute Administratorkonto automatisch an. Jetzt bietet es sich an, alle ausstehenden Updates einzuspielen und danach, wenn nötig, den virtuellen Rechner neu zu starten. Dieser bootet dann wieder automatisch in Audit Mode.



Im Audit Mode kann man die aktuellen Updates in das Image integrieren

Anschließend installiert man die Software, welche im Image enthalten sein soll, entfernt bzw. fügt Features hinzu oder passt die Installation nach sonstigen Erfordernissen an.

Store Apps (offline) aus Windows-Image entfernen

Ein gängiges Anliegen besteht darin, die von Microsoft installierte Crapware in Form von Store Apps zu entfernen. Allerdings sollte man sie nicht pauschal abräumen, weil sich unter ihnen auch systemrelevante Apps befinden.

Mittlerweile finden sich unter den Store Apps auch solche von Bedeutung, die man nicht unbesehen löschen sollte. So umfasst der Desktop App Installer nicht nur *winget,* sondern auch die Funktionen für die *MSIX-Installation.*

Auch das Löschen der App für den Microsoft Store war nie empfehlenswert, aber selbst gegen das Blockieren des Stores spricht, dass sie für die Installation von Systemkomponenten benötigt wird. Das trifft etwa auf die Distributionen im Subsystem für Linux zu, aber auch auf Schriftarten oder Sprachpakete.

Darüber hinaus liegen ehemalige Win32-Programme aus dem Zubehör jetzt als Store Apps vor, zum Beispiel Notepad oder der Taschenrechner. Und diese möchte man zumeist ebenfalls nicht eliminieren.

Bereitstellung versus Installation

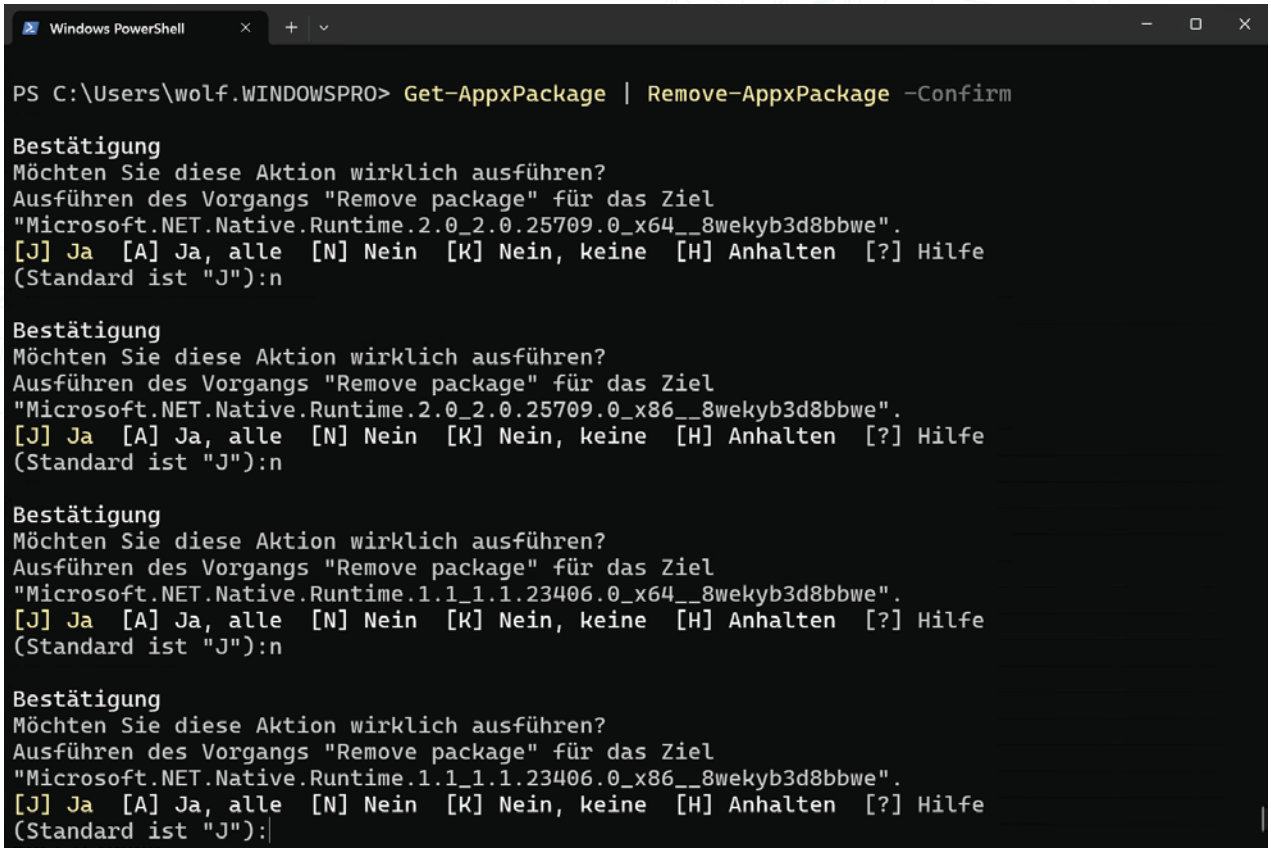
Eine Eigenheit der Store Apps besteht darin, dass sie in Windows erst systemweit bereitgestellt werden und jeder Benutzer eine eigene Kopie davon erhält, wenn er sie installiert. Beim Anpassen eines Images für das Deployment muss man daher die bereitgestellten Apps entfernen, um zu verhindern, dass User diese bei ihrer ersten Anmeldung erhalten.

Wenn man eine Referenzinstallation von Windows erstellt, die als Custom Image erfasst werden soll, dann muss man für den angemeldeten Administrator auch die installierte Version der betreffenden Apps entfernen, bevor man die bereitgestellten Pakete entfernt. Andernfalls erhält man eine Fehlermeldung beim Generalisieren des Abbilds mit Sysprep.

Diese Aufgabe erfüllt Remove-AppxPackage, das man so aufrufen kann:

```
Get-AppxPackage | Remove-AppxPackage -Confirm
```

Der Befehl fragt für jedes Package nach, ob es entfernt werden soll.



Für den aktuellen User installierte Store Apps selektiv entfernen

Bereitgestellte Apps entfernen

Unabhängig davon, ob man die bereitgestellten Apps aus einem laufenden Windows oder offline entfernen möchte, das dafür zuständige Cmdlet ist das gleiche. Im ersten Fall verwendet man den Schalter *Online*, im zweiten spezifiziert man den Pfad zum Image.

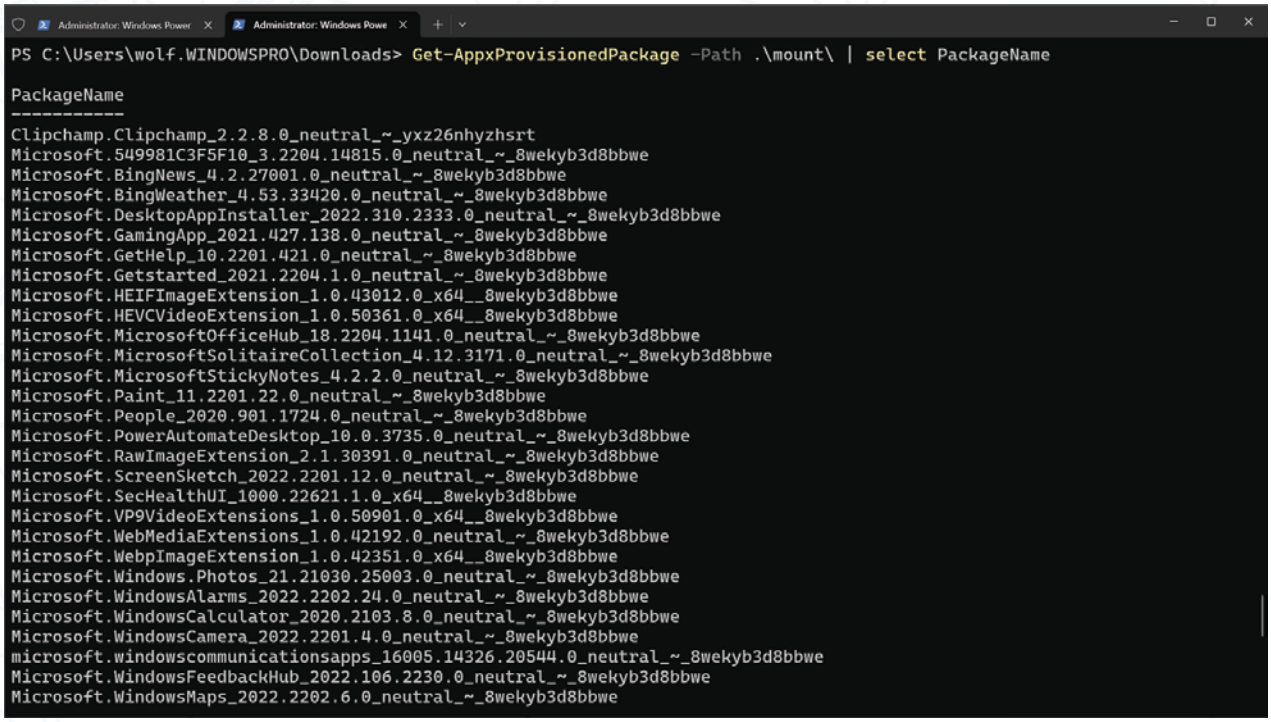
Mit diesem Befehl gibt man den *PackageName* der im Abbild bereitgestellten Apps aus:

Get-AppxProvisionedPackage -Online | Select PackageName

Für ein Offline-System würde der Aufruf dagegen so aussehen:

Get-AppxProvisionedPackage -Path .\mount\ | Select PackageName

Anhand dieser Liste kann man sich einen Überblick verschaffen, welche Apps man entfernen möchte.



PackageName der im Image bereitgestellten Apps anzeigen

Deren *PackageName* kopiert man in eine Textdatei, und zwar einen Namen pro Zeile, und speichert sie zum Beispiel als *delapps.txt*.

Liste mit Namen von bereitgestellten Packages, die man entfernen möchte

Anschließend löscht man die darin enthaltenen Apps in einem Image folgendermaßen:

```
Get-Content .\delapps.txt |

foreach{

Remove-AppxProvisionedPackage -PackageName $_ -Online
# Aufruf für Offline-Image:
#  Remove-AppxProvisionedPackage -PackageName $_ -Path .\mountpath
}
```

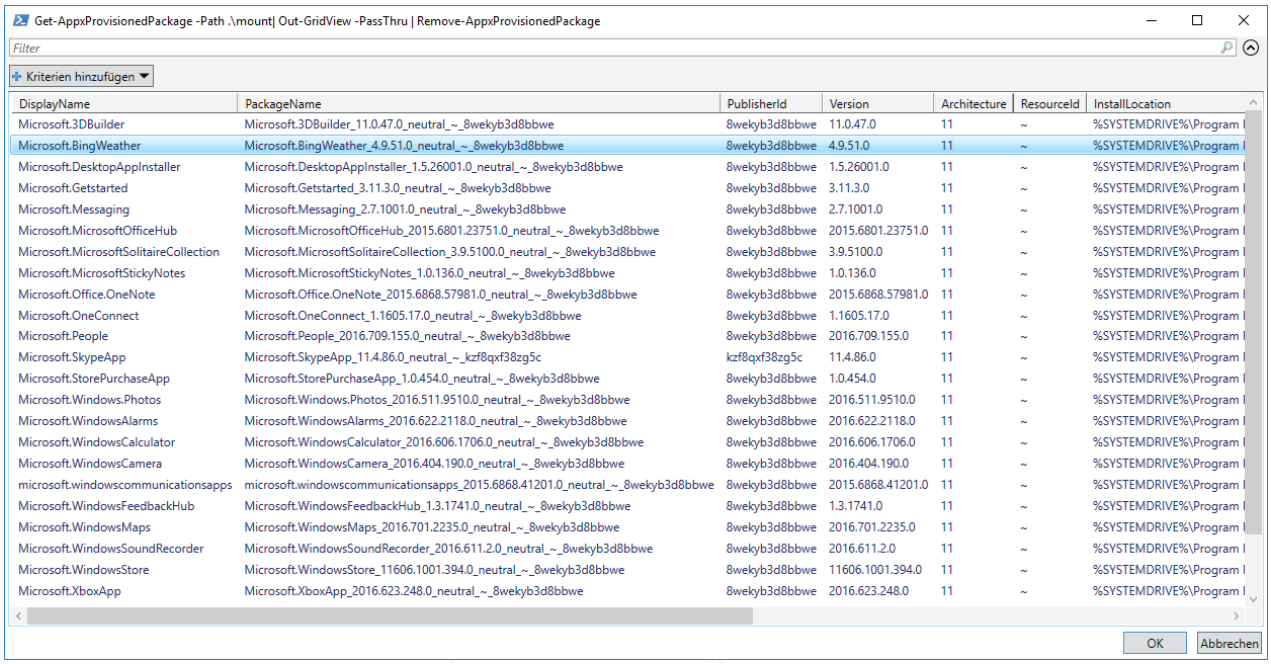
Apps mit Out-GridView auswählen

Eine elegantere Lösung lässt sich mit Out-GridView realisieren. Dabei lädt man die Apps in eine Tabelle, markiert die gewünschten Einträge und startet den Löschvorgang durch Klicken auf *Ok*:

```
Get-AppxProvisionedPackage -Online | Out-GridView -PassThru |
Remove-AppxProvisionedPackage
```

Für ein Offline-Image würde der Befehl so aussehen:

```
Get-AppxProvisionedPackage -Path .\mountpath | Out-GridView -PassThru |
Remove-AppxProvisionedPackage
```



Apps selektiv aus einem Image löschen mit Out-GridView

Antwortdatei integrieren

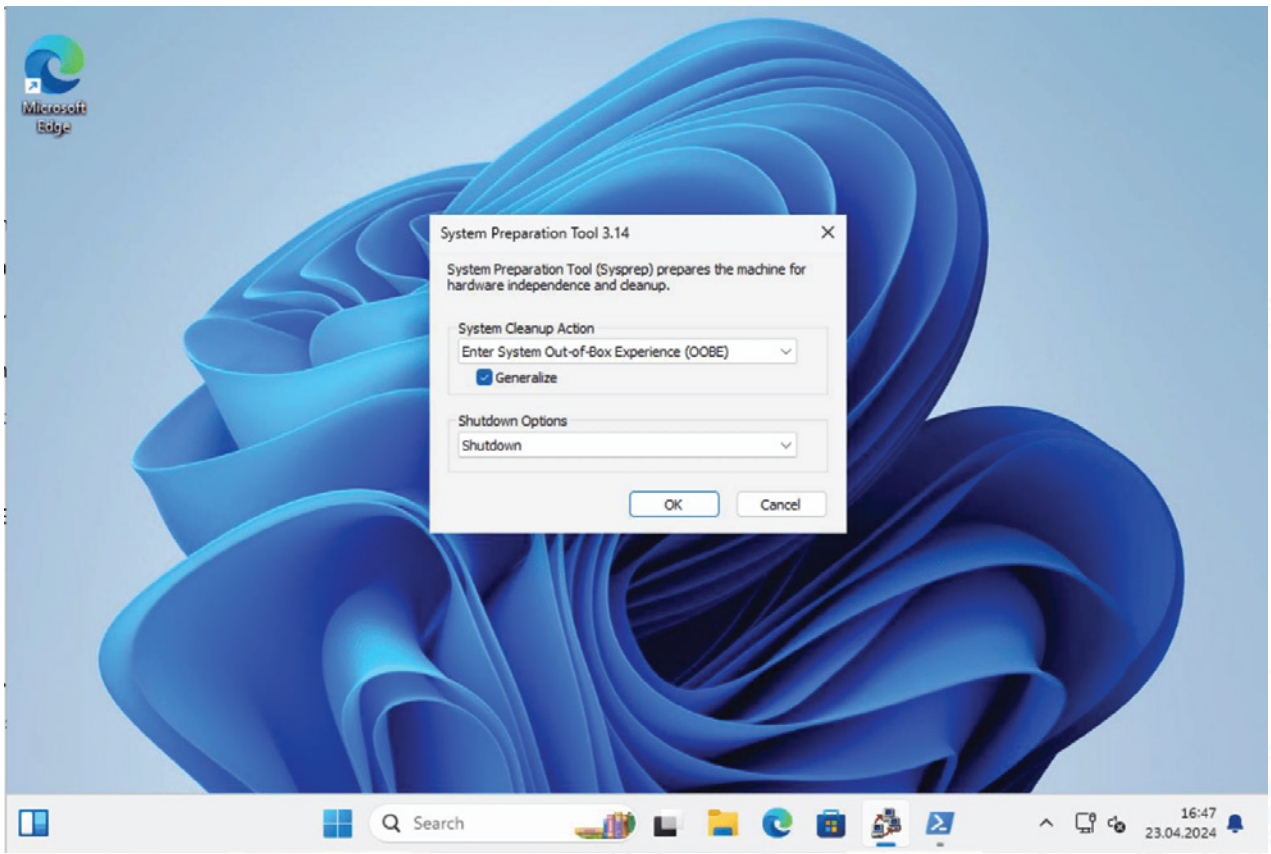
Wenn man die spätere Installation auf den Clients mit einer Antwortdatei automatisieren möchte, zum Beispiel um die Dialoge der OOBE-Phase zu überspringen, dann kopiert man die Datei *unattend.xml* nach *%SystemRoot%\system32\panther*.

Schließlich kann man bei Bedarf noch Windows-Einstellungen konfigurieren, wie sie auf den Ziel-PCs gewünscht sind.

Installation mit Sysprep generalisieren

Falls man diese Referenzinstallation weiterhin verwenden möchte, um daraus angepasste Images zu generieren, sollte man an dieser Stelle einen Snapshot anlegen, zu dem man nach dem Generalisieren wieder zurückkehren kann.

Zum Abschluss wechselt man zu sysprep, das im Audit-Modus automatisch startet (ansonsten findet man es unter %SYSTEMROOT%\system32\sysprep). Beim Start ohne Parameter erhält man eine GUI, über die man unter *Systembereinigungsaktion* ("System Cleanup Action") die Option *Out-of-Box-Experience (OOBE)* auswählt und die Checkbox *Verallgemeinern* ("Generalize") anhakt.



Im Audit Mode startet sysprep automatisch

Bei den Optionen für das Herunterfahren aktiviert man *Herunterfahren* ("Shutdown") und klickt auf OK.

Image erfassen

Nachdem die VM ausgeschaltet ist, startet man sie über ein anderes Medium, um das Image als WIM-Archiv erfassen zu können. Außerdem muss man überlegen, wo man das Image speichern möchte.

Dafür bietet sich eine zusätzliche VHDX an, die man nach dem erneuten Herunterfahren der VM im Host-OS mounten kann. Daraus entnimmt man dann das WIM-Abbild. Alternativ kommt dafür natürlich auch ein Netzlaufwerk in Frage.

Virtuelle Maschine von Setup-ISO starten

Wenn man Windows von der Microsoft-ISO installiert hat und diese noch als virtuelle DVD in die VM eingehängt ist, dann kann man Zeit sparen, indem man sie gleich als Boot-Medium nutzt. Dazu stellt man sicher, dass sie in der Boot-Reihenfolge noch an erster Stelle steht.

Dann startet man die VM vom DVD-Laufwerk und drückt Umschalten + F10, sobald der erste Setup-Dialog erscheint. Dies öffnet die Kommandozeile eines rudimentären WinPE, das für die Ausführung von DISM reicht.

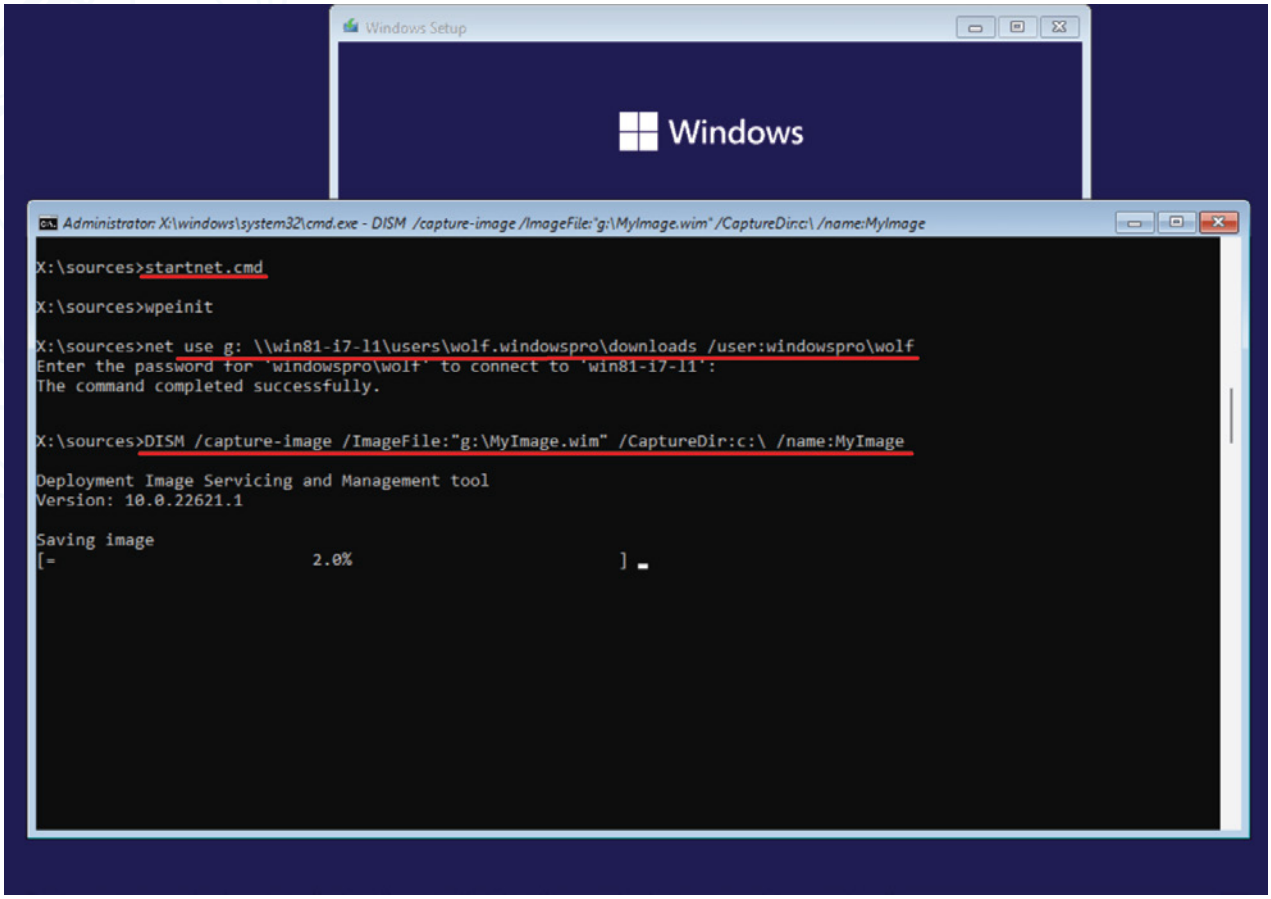
Wenn man die WIM auf einem Netzlaufwerk speichern möchte, dann führt man zuerst den Befehl startnet.cmd

aus, um das Netzwerk zu initialisieren und verbindet dann ein Share nach diesem Muster:
net use g: \\server\verzeichnis /user:contoso\user

Anschließend verwendet man DISM, um das Image zu erfassen:

Dism /Capture-Image /ImageFile:G:\MyImage.wim /CaptureDir:C:\ /ScratchDir:G:\ /Name:MyImage

In diesem Beispiel schreibt DISM den Inhalt des Laufwerks C: (CaptureDir) nach G:\MyImage.wim (ImageFile). Das *ScratchDir* benötigt man, weil auf dem WinPE-Laufwerk nicht genügend Speicherplatz für diese Operation vorhanden ist und man daher ohne diesen Parameter einen "Error: 6" und die Meldung "The handle is invalid" kassiert".



WinPE über die Setup-ISO starten, Netzlaufwerk verbinden und Image mit DISM erfassen

Image mit PowerShell aus WinPE erfassen

Hat man jedoch ein Boot-Medium mit Windows PE eingerichtet, dann verbindet man das virtuelle DVD-Laufwerk mit der WinPE-ISO und bootet die VM von dort.

Wenn dort PowerShell verfügbar ist, dann gibt man folgende Kommandos ein, andernfalls geht man wie oben beschrieben vor und verwendet DISM. Zuerst ordnet man dem gewünschten Network-Share einen Laufwerks¬buchstaben zu:

```
New-SmbMapping -LocalPath G: -RemotePath "\\server\pfad" -UserName "contoso\me" `
-Password "P@ssw0rd"
```

Anschließend erfasst man das Systemabbild:

New-WindowsImage -ImagePath "G:\MyImage.wim" -CapturePath "C:\" `

-ScratchDirectory "G:\" -Name "MyImage"

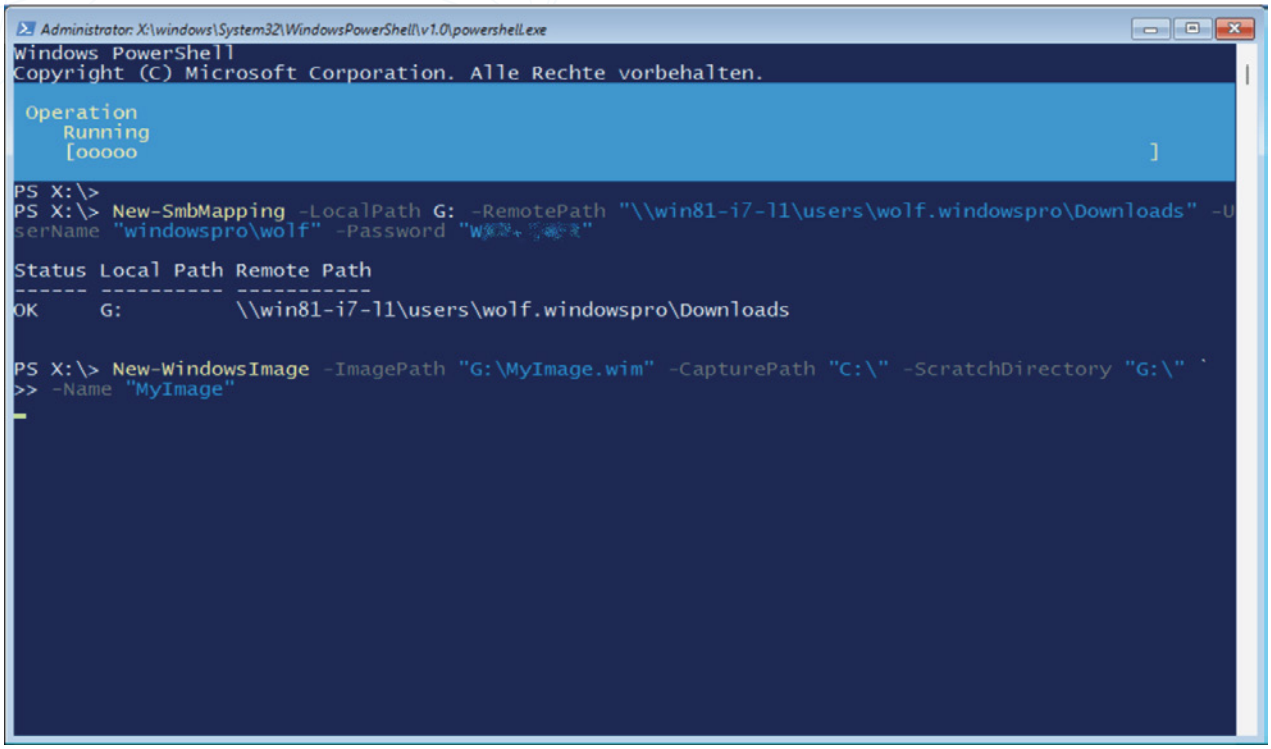


Image aus Windows PE mit PowerShell erfassen und auf einem Network Share speichern

Nun sollte die angepasste Windows-Installation als WIM vorliegen. Diese kann man nun zurück in die Setup-ISO schreiben, auf einem WDS-Server hinterlegen oder mit Hilfe von DISM anwenden.

Updates (.msu) Offline -Image integrieren

Windows-Images müssen vor dem Deployment regelmäßig auf den neuesten Stand gebracht werden. Dabei ist es egal, ob man ein individuell angepasstes Abbild oder install.wim von der Microsoft-ISO verwendet. Mit dem DISM-Modul kann man in PowerShell Updates, Apps und Treiber in ein Image einspielen, ohne dass man dieses booten und nach dem Hinzufügen der Updates erneut erfassen muss.

Ein Nachteil der Offline-Wartung ist jedoch, dass sie keine im Windows-Image installierten Anwendungen aktualisieren kann.

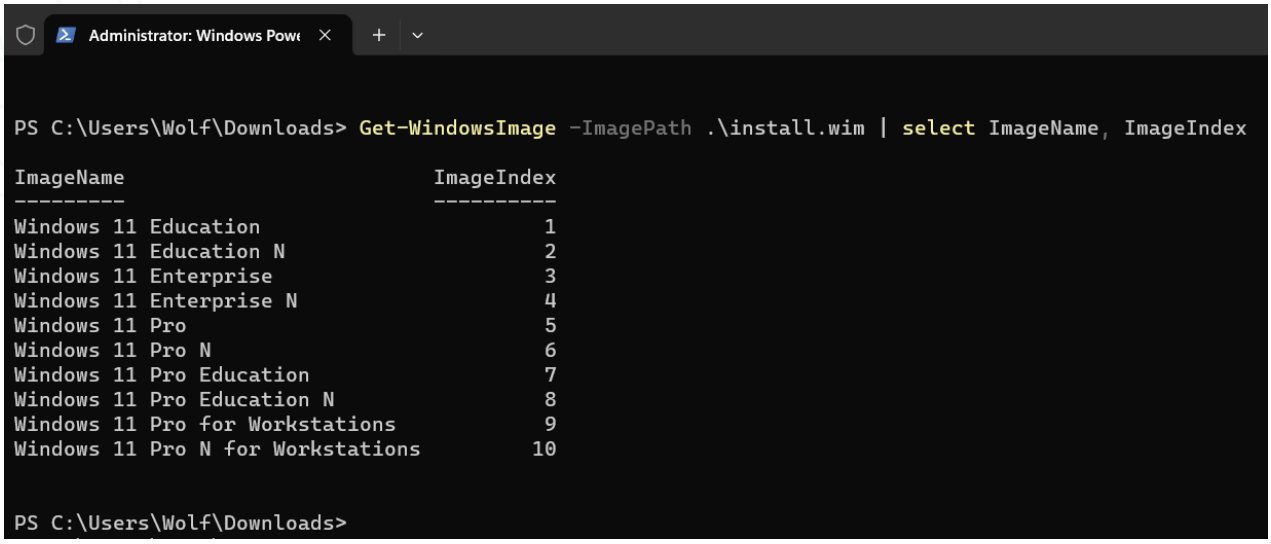
In unserem Beispiel verwenden wir die unmodifizierte install.wim, aber das Vorgehen ist für alle WIM-Archive gleich. Man findet die Datei auf Microsofts ISO im Verzeichnis *sources*.

Index der Images aus WIM-Archiv auslesen

Für das Offline-Servicing von Images sehen die Bordmittel DISM bzw. die Cmdlets aus dem entsprechenden PowerShell-Modul vor. Letztere lassen sich wesentlich einfacher nutzen als das unhandliche Dienstprogramm mit seiner inkonsistenten Syntax.

Im ersten Schritt zeigt man den Inhalt des WIM-Archivs an, da dieses mehrere Images enthalten kann und man den Index des Abbilds benötigt, das man mounten möchte:

Get-WindowsImage -ImagePath .\install.wim | select ImageName, ImageIndex



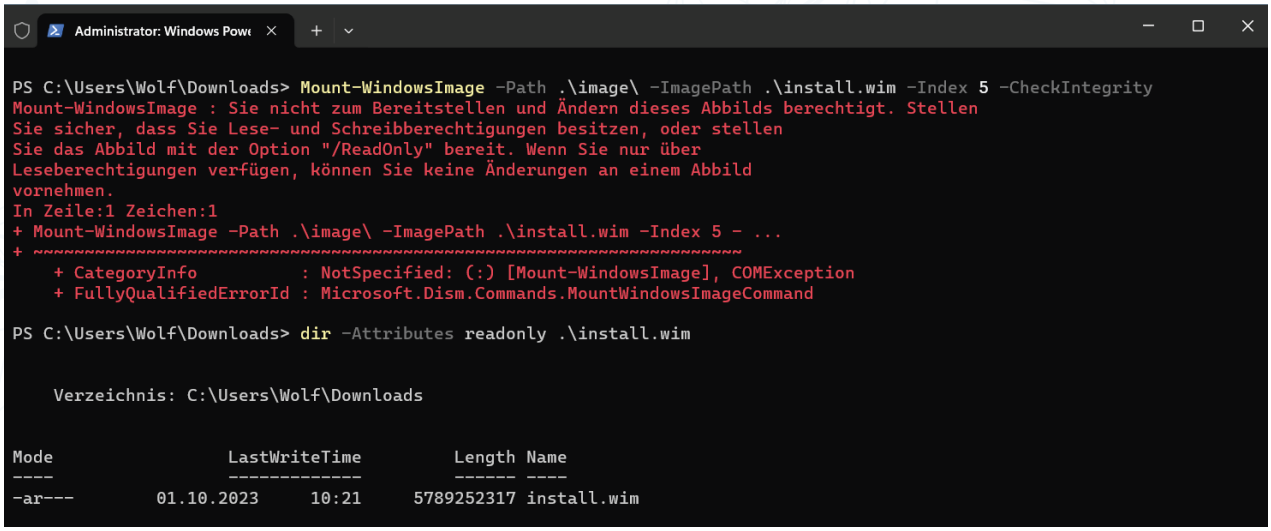
Inhalt eines WIM-Archivs anzeigen mit Get-WindowsImage

Image mounten

Nachdem man den Index ermittelt hat, kann man das Image für das Offline-Servicing bereitstellen:
Mount-WindowsImage -Path .\image\ -ImagePath .\install.wim -Index 5 -CheckIntegrity

Dieser Aufruf mountet das Image mit dem Index 5 (Windows 11 Pro) in das Unterverzeichnis .\image und prüft dabei dessen Integrität. Hat man die WIM-Datei gerade aus der Installations-ISO entnommen, dann läuft man in folgende Fehlermeldung:

Mount-WindowsImage: Sie sind nicht zum Bereitstellen und Ändern dieses Abbilds berechtigt. Stellen Sie sicher, dass Sie Lese- und Schreibberechtigungen besitzen, oder stellen Sie das Abbild mit der Option "/ReadOnly" bereit.



Das Mounten von install.wim scheitert am Readonly-Attribut der Datei

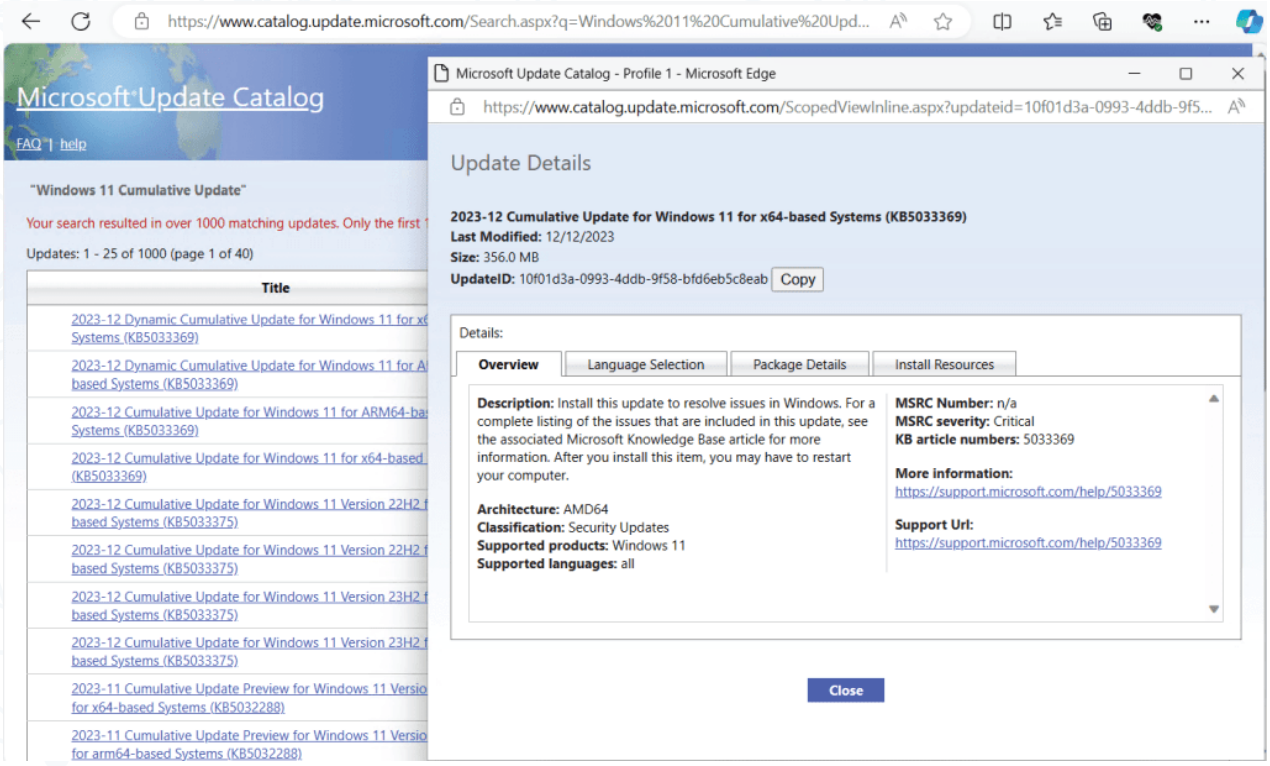
Auf Englisch lautet sie:

Mount-WindowsImage: You do not have permissions to mount and modify this image. Verify that you have Read/Write permissions or mount the image using the /ReadOnly option.

Die Meldung ist irreführend, weil das Problem nicht in mangelnden Berechtigungen besteht, sondern durch das Nur-Lesen-Attribut der WIM-Datei verursacht wird. Dieses entfernt man mit

attrib -r install.wim

Nun sollte im nächsten Anlauf das Mounten des Images erfolgreich verlaufen, so dass man sich an das Einspielen der Updates machen kann. Diese wird man in der Regel im .msu- oder .cab-Format vom Update Catalog herunterladen.



Fehlende Patches aus dem Microsoft Update Catalog herunterladen

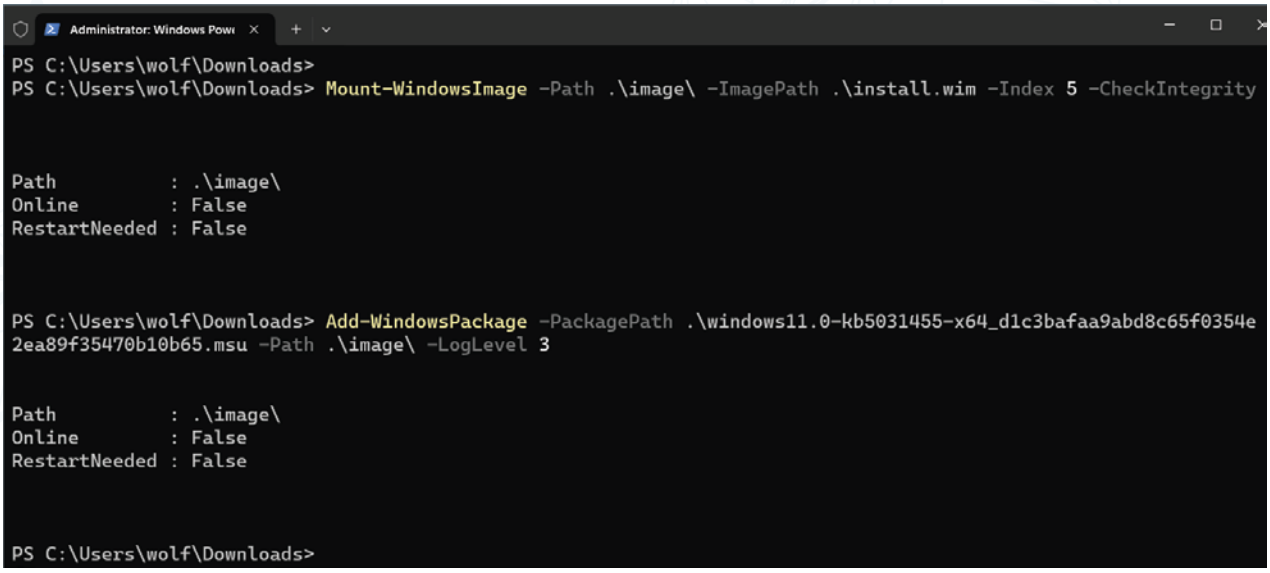
Updates in das Image integrieren

In unserem Beispiel handelt es sich um ein kumulatives Update für Windows 11 x64. Dieses integriert man mit folgendem Befehl in das Abbild:

```
Add-WindowsPackage -PackagePath .\windows11.0-kb5031455-x64_[...].msu -Path .\image\ -LogLevel 3
```

Dieser Aufruf enthält den Pfad zum Update sowie zum Verzeichnis, in welches wir das Image gemountet haben. Wenn man für den Parameter *PackagePath* ein Verzeichnis anstatt einer Datei angibt, dann installiert das Cmdlet alle dort gespeicherten .msu- und .cab-Dateien in das Image.

Der Loglevel von 3 schreibt Fehler, Warnungen und Informationen in das Logfile (standardmäßig handelt es sich dabei um dism.log in %WINDIR%\Logs\Dism).



Update in das Offline-Image integrieren

Neben Updates kann man mit Add-WindowsCapability und Add-WindowsDriver nach dem gleichen Muster optionale Features bzw. Treiber in das Image installieren. Um Store-Apps hinzuzufügen, die automatisch für jeden neuen User auf dem Rechner eingerichtet werden, verwendet man Add-AppxProvisionedPackage.

Zum Abschluss kann man überprüfen, ob alle gewünschten Pakete im Image enthalten sind:

Get-WindowsPackage -Path .\image\ | sort InstallTime -Descending | more

Dieser Aufruf sortiert die installierten Pakete nach dem Installationsdatum, und zwar in absteigender Reihenfolge, so dass die zuletzt hinzugefügten Packages zuerst angezeigt werden.

Image aushängen

Schließlich kann man das aktualisierte Image speichern. Dafür verwendet man entweder Save-WindowsImage und hängt das Abbild danach aus oder man dismountet es gleich und speichert es bei dieser Gelegenheit:

Dismount-WindowsImage -Path .\image\ -Save -CheckIntegrity

In beiden Fällen kann man den Schalter CheckIntegrity nutzen, um die Integrität des Images zu prüfen.

Images mit OSDBuilder automatisch offline aktualisieren

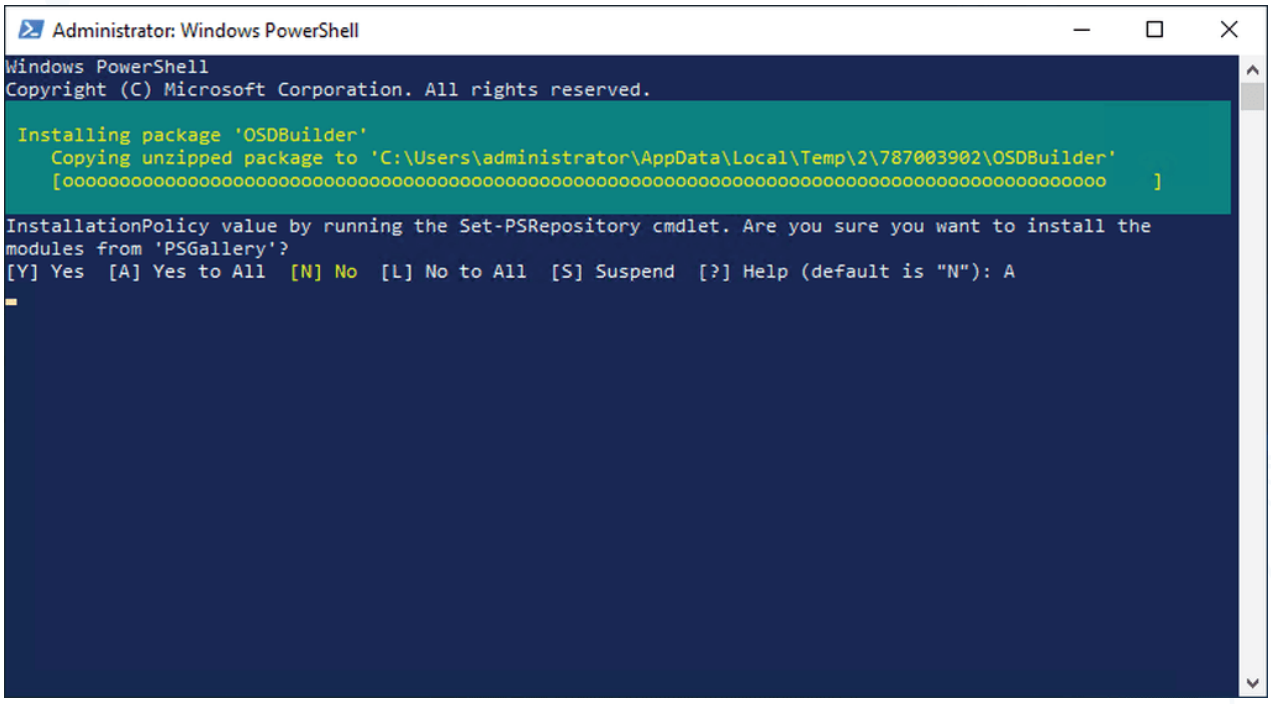
OSDBuilder ist ein PowerShell-Modul, das der Offline-Wartung von Windows-Images dient. Es automatisiert die oben beschriebenen Schritte, welche beim Einsatz der Bordmittel erforderlich sind.

Das Tool erstellt eine Antwortdatei, die als *Task* bezeichnet wird. Dieser zeichnet alle Informationen auf, die für die Aktualisierung des Windows-Images erforderlich sind, und es ist keine Interaktion mit dem Betriebssystem im Image nötig. Man kann auch mehrere OSDBuilder-Tasks nacheinander ausführen.

OSDBuilder installieren

OSDBuilder ist einfach zu installieren, weil es sich dabei um ein PowerShell-Modul handelt:

```
Install-Module OSDBuilder
```



Importieren des PowerShell-Moduls OSDBuilder

Danach muss man bei Bedarf die Execution-Policy anpassen und das Modul importieren:

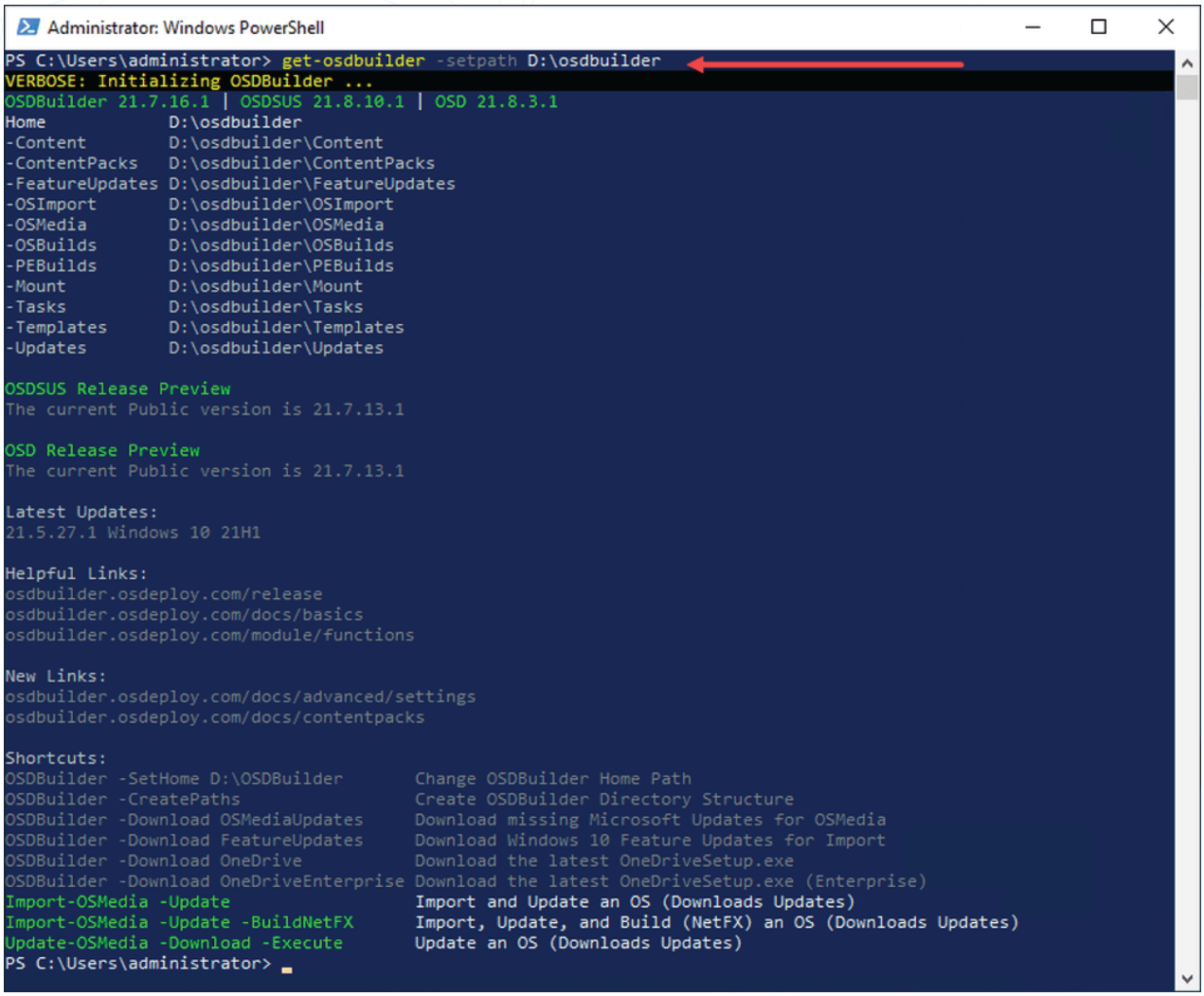
```
Set-ExecutionPolicy Bypass
```

```
Import-Module OSDBuilder
```

Konfigurieren der Ordnerstruktur für OSDBuilder

Mit diesem Befehl legt man den Zielordner für Image-Builds fest:

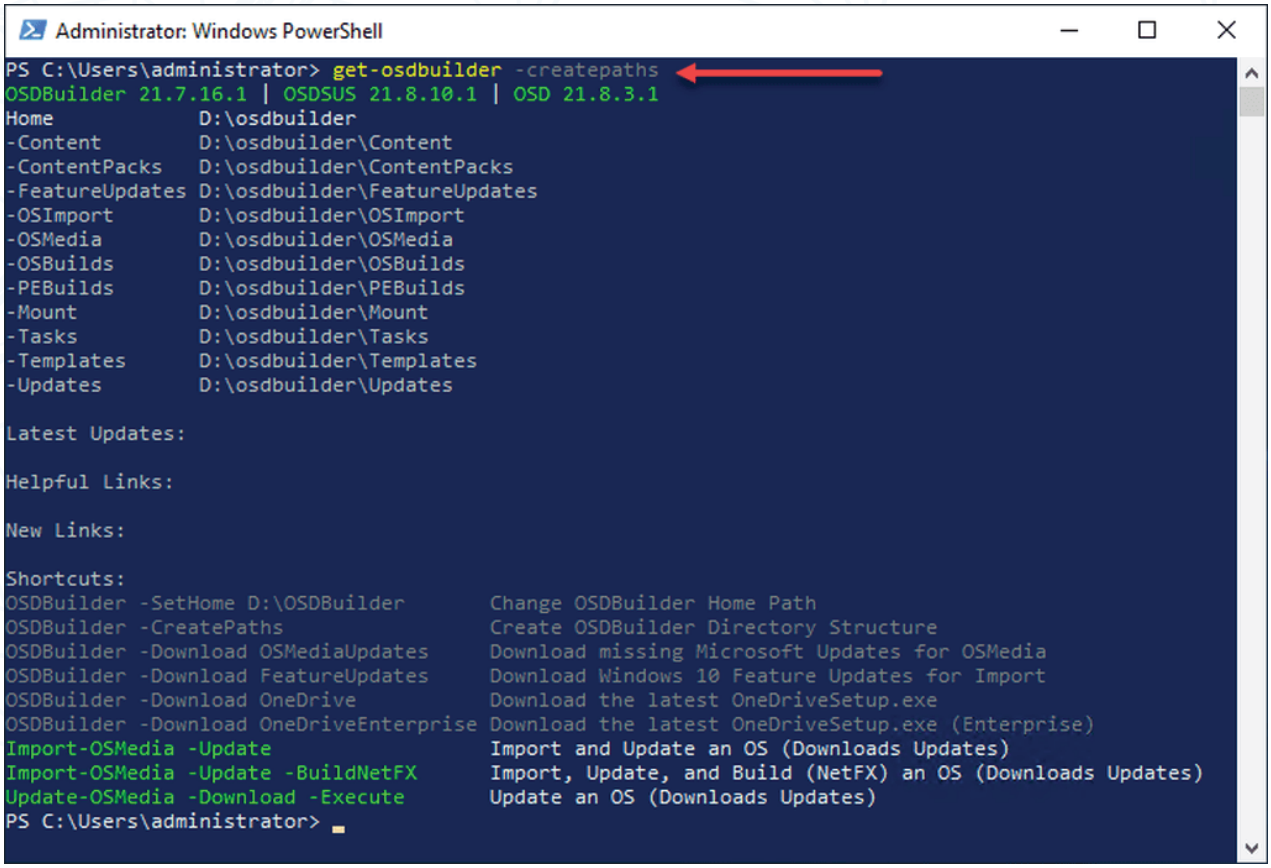
Get-OSDBuilder -SetPath <Pfad-für-Images>



OSDBuilder-Pfad festlegen

Danach füllt man die Ordnerstruktur im oben festgelegten Zielpfad mit den benötigten Unterverzeichnissen:

Get-OSDBuilder -CreatePaths

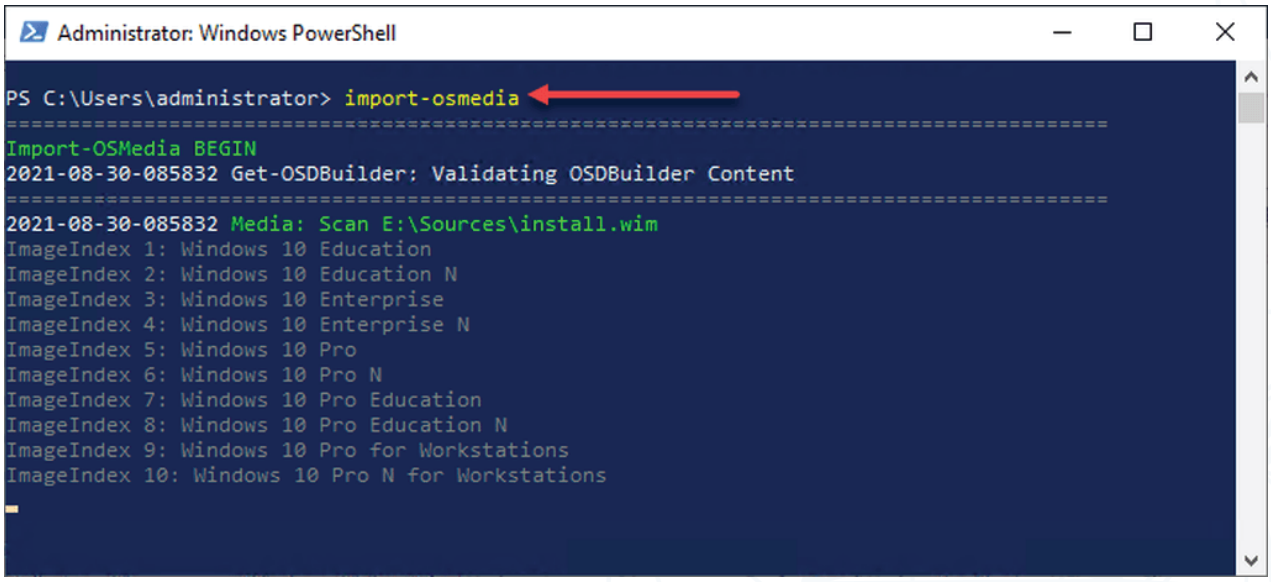


Pfade im OSDBuilder-Ordner erstellen

Installationsmedien importieren

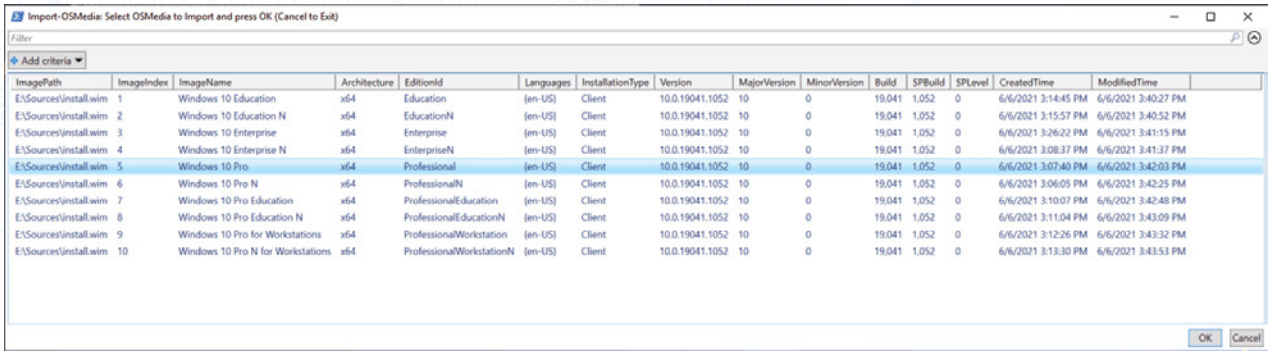
Als Nächstes müssen wir die OS-Medien importieren, die wir aktualisieren und warten wollen:

Import-OSmedia



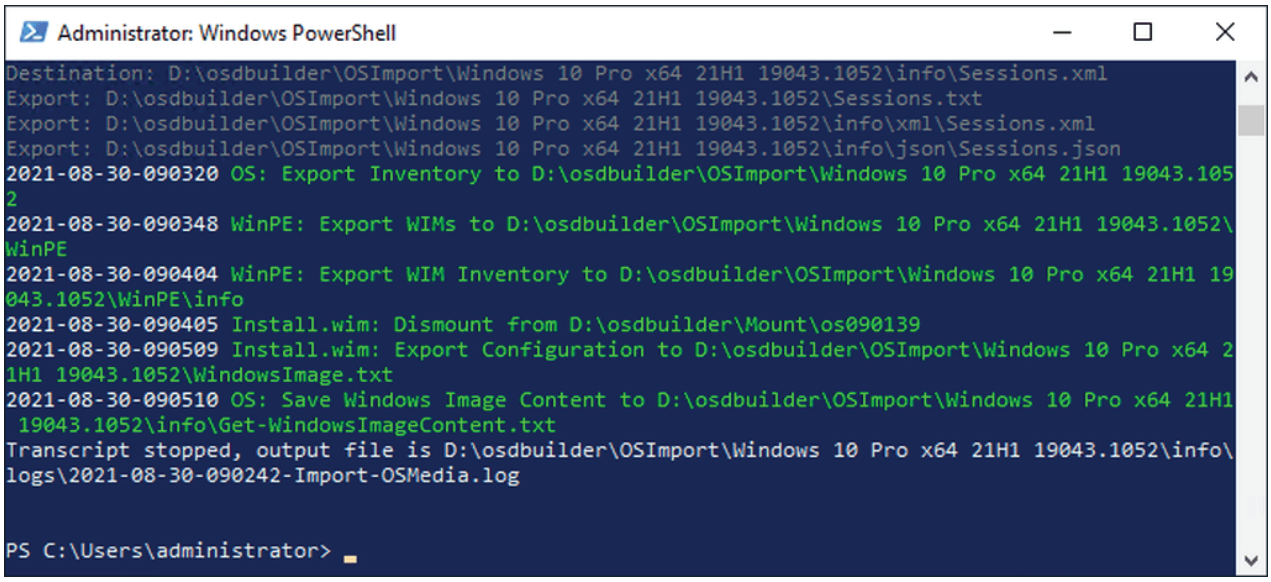
Installationsmedien von Windows 10 in OSDBuilder importieren

Aus der folgenden Liste wählt man die Edition aus, die man für die benutzerdefinierte Image-Bereitstellung verwenden möchte. Die ISO für Windows 10 enthält mehrere Abbilder, so dass man sich für eines davon entscheiden muss.



Gewünschte Edition auswählen

Der Vorgang startet und die Medien werden in den OSDBuilder-Ordner importiert. Dafür wird auch ein Ausgabeprotokoll erstellt.

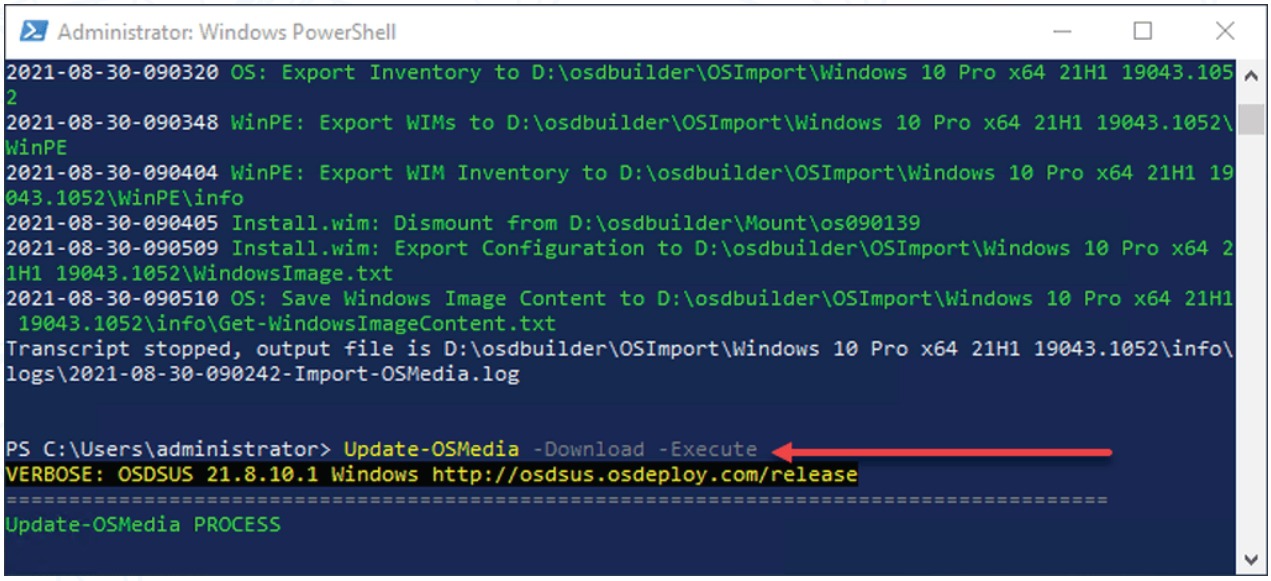


Windows-Abbild aus der ISO in OSDBuilder importieren

OS-Medien aktualisieren

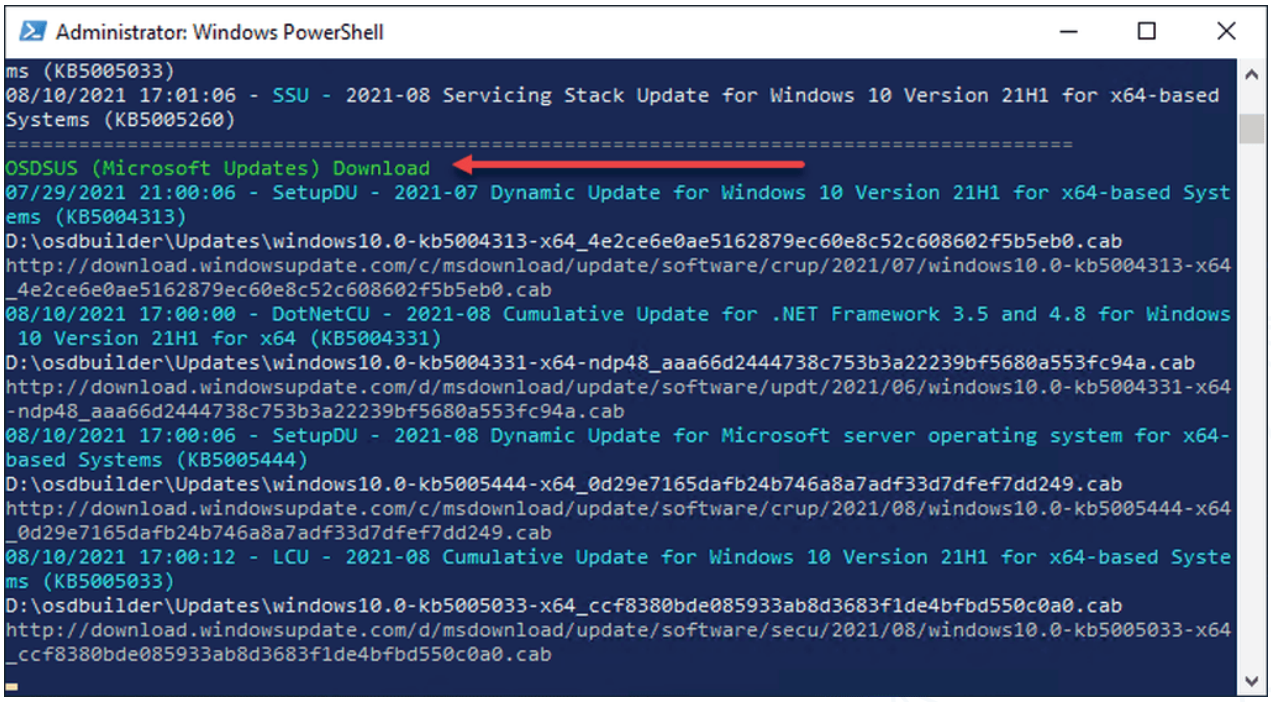
Zum eigentlichen Aktualisieren unseres Windows-Abbilds verwenden wir folgenden Befehl:

```
Update-OSMedia -Download -Execute
```



OSDBuilder lädt die Updates herunter und aktualisiert damit die Installationsmedien

Hier wählt man das Image aus der Liste aus, das aktualisiert werden soll. OSDBuilder beginnt nun damit, die Updates herunterzuladen und in das Abbild zu integrieren.



OSDBuilder lädt die Updates herunter und aktualisiert damit die Installationsmedien

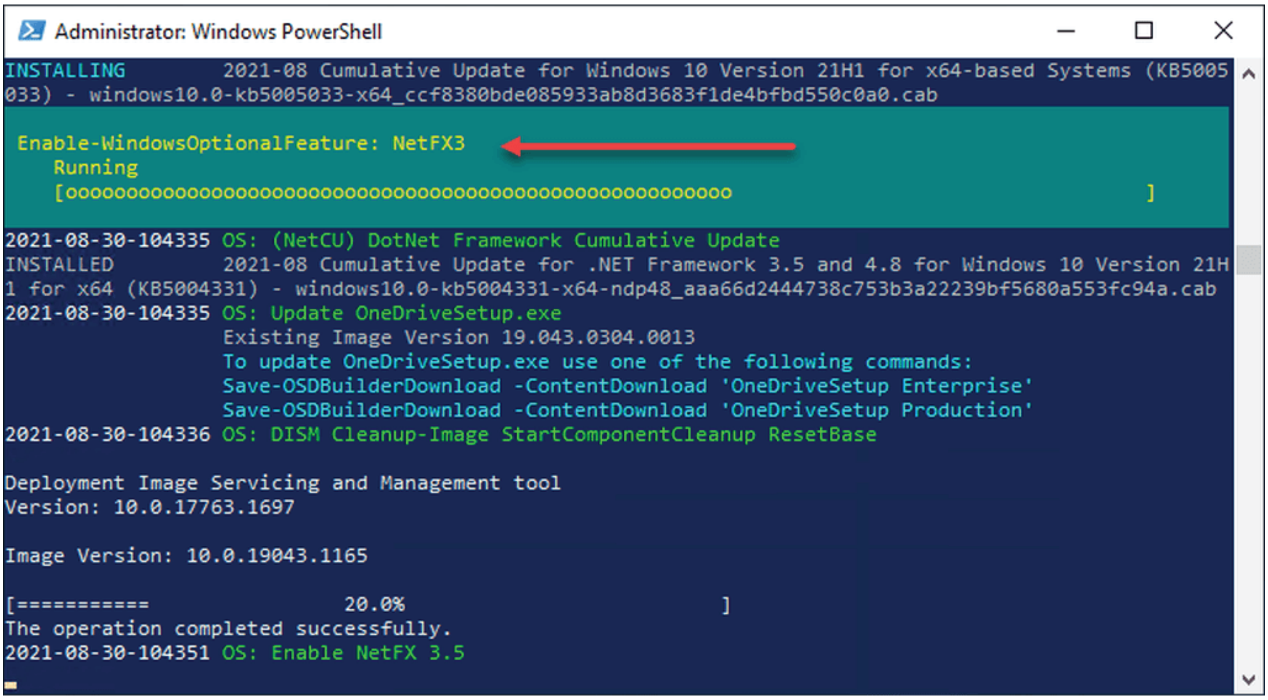
OSBuild-Task erstellen

Wie bereits erwähnt, verwendet OSDBuilder Tasks, um die Offline-Wartung des Images zu automatisieren. Um eine solche Aufgabe zu erstellen und beispielsweise die Windows-Komponente .NET3.5 SP1 zu hinzuzufügen, führt man diesen Befehl aus:

New-OSBuildTask -TaskName <Task-Name> -EnableFeature <Feature, das man installieren möchte>

Anschließend erscheint eine Liste von Images, aus der man das gewünschte auswählt. Um den neuen Task auszuführen, gibt man folgenden Befehl ein:

New-OSBuild -Download -Execute



Die Update-Task fügt das benötigte Feature hinzu

Bootfähige ISO für Windows-Image (wim) erstellen

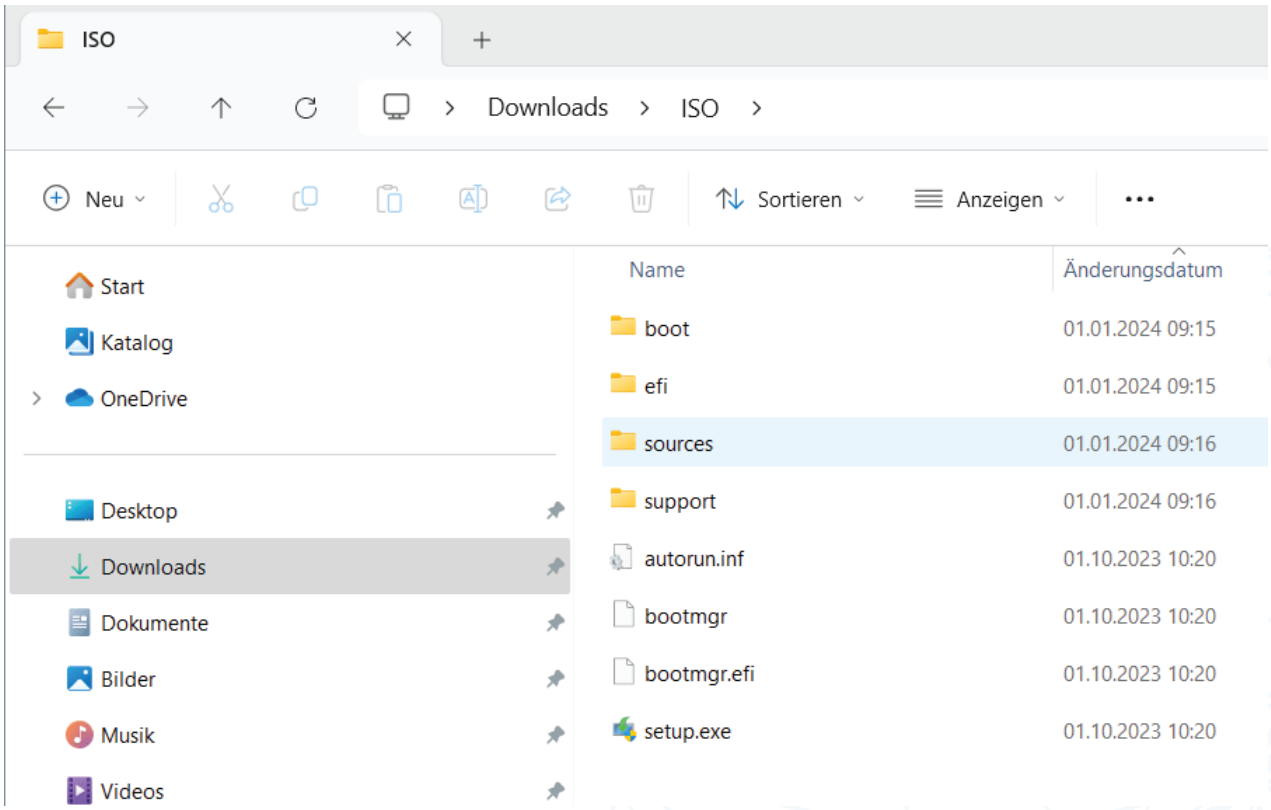
Wenn man ein Windows-Image mit den neuesten Updates aktualisiert und eventuell noch zusätzlich angepasst hat, dann kann man es zur Installation auf den Clients bereitstellen. Verwendet man die Windows Deployment Services (WDS) für die OS-Installation, dann lädt man dort nur die WIM-Archive hoch.

Bootet man die Rechner für die Lite Touch Installation von einem lokalen Datenträger, dann wird man dafür heutzutage meist einen bootfähigen USB-Stick mit Windows PE verwenden.

Die Installation direkt von ISO ist dagegen besonders für virtuelle Maschinen interessant, weil man diese davon booten kann.

Angepasste WIM-Datei übernehmen

Um ein ISO-Installationsmedium mit einem angepassten WIM-Image zu erstellen, mountet man die Original-DVD und kopiert deren Inhalt in ein Verzeichnis. Anschließend ersetzt man im Ordner *sources* die ursprüngliche install.wim durch die individuell angepasste Version.



Das WIM-Archiv mit den Installationsdateien befindet sich im Verzeichnis sources

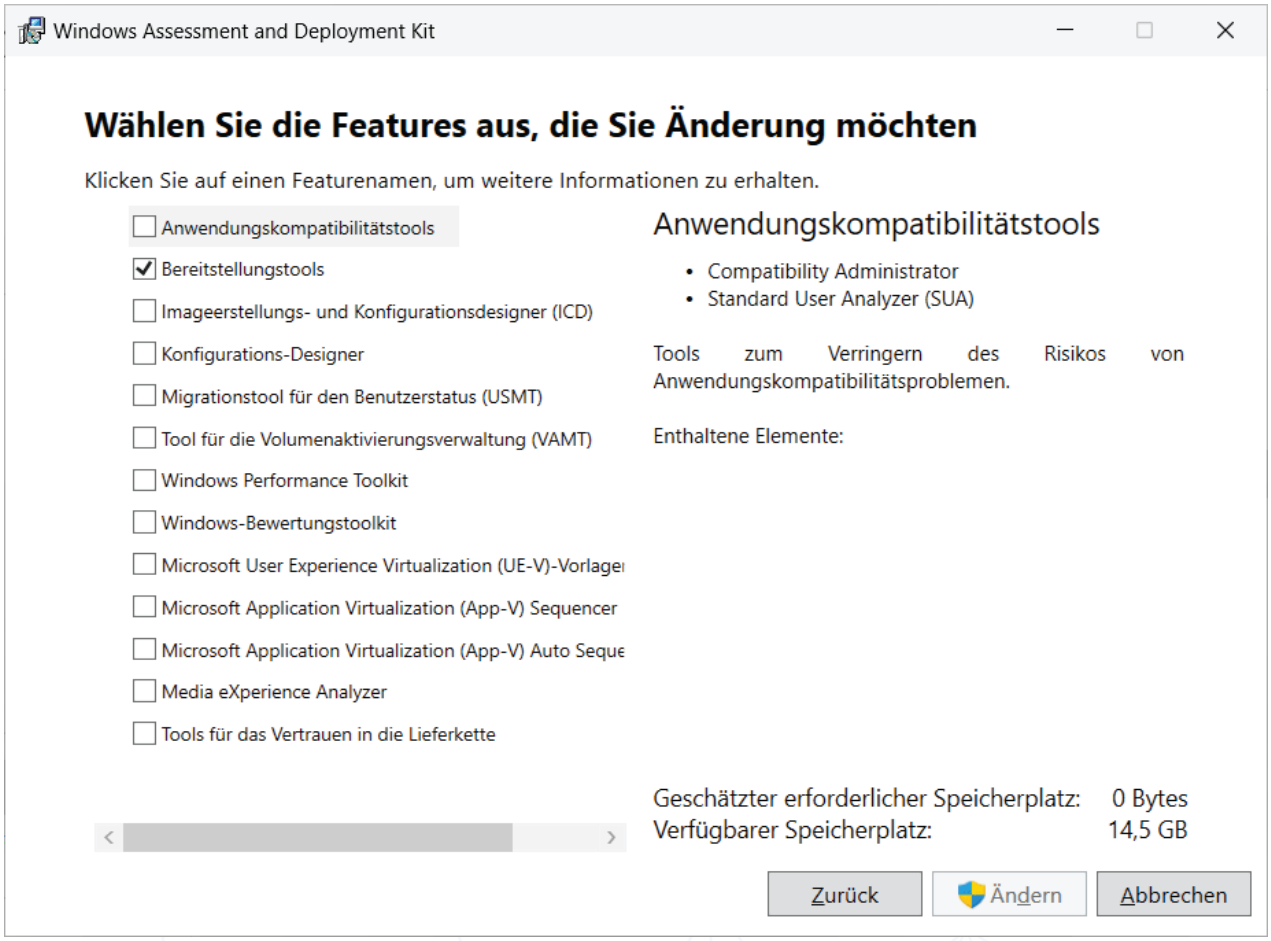
Nun muss man diesen Verzeichnisbaum wieder in eine bootfähige ISO-Datei schreiben. Grundsätzlich gibt es mehrere kostenlose Tools für das Erstellen einer ISO aus einem Ordner, die zumeist auch eine grafische Oberfläche bieten. Viele davon wurden jedoch schon länger nicht mehr aktualisiert.

ISO mit oscdimg erzeugen

Nachdem das Windows ADK mit *oscdimg.exe* ein eigenes Werkzeug für diesen Zweck umfasst, muss man seine Zeit nicht darauf verwenden, ein brauchbares Tool unter den kostenlosen Angeboten zu suchen.

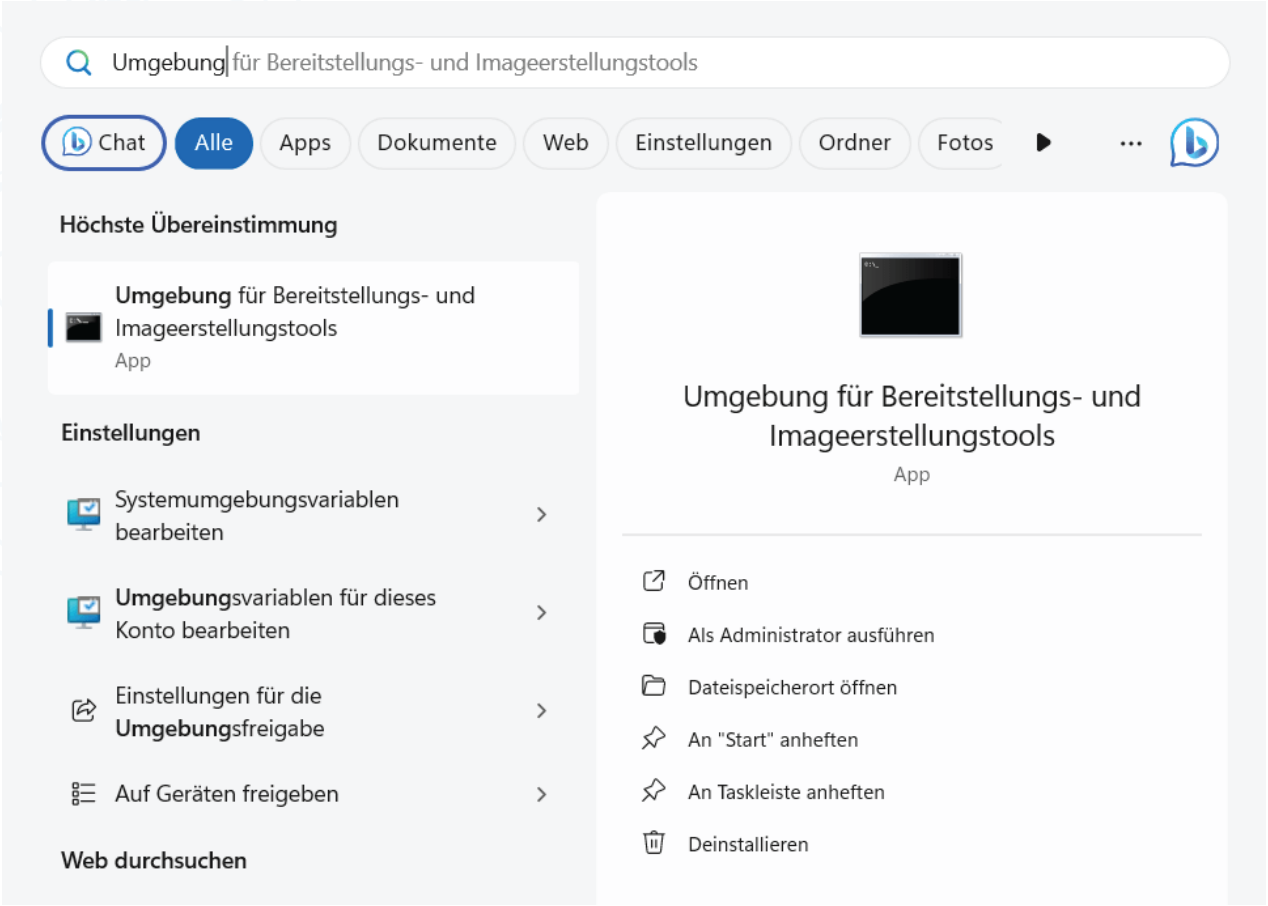
Der Nachteil von *oscdimg.exe* besteht allerdings darin, dass es sich dabei um ein Konsolenprogramm mit zahlreichen Optionen handelt. Viele davon sind aber mittlerweile nicht mehr relevant, so dass man sich auf eine Handvoll Schalter beschränken kann.

Das Tool befindet sich nach der Installation der ADK-Bereitstellungs-Tools im Verzeichnis *%ProgramFiles(x86)%\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\amd64\Oscdimg*. Der Ordner enthält zudem die Dateien für den Boot-Sektor, und zwar für UEFI (efisys.bin) und BIOS (Etfsboot.com).



Die Installation der Bereitstellungstools umfasst auch oscdimg.exe

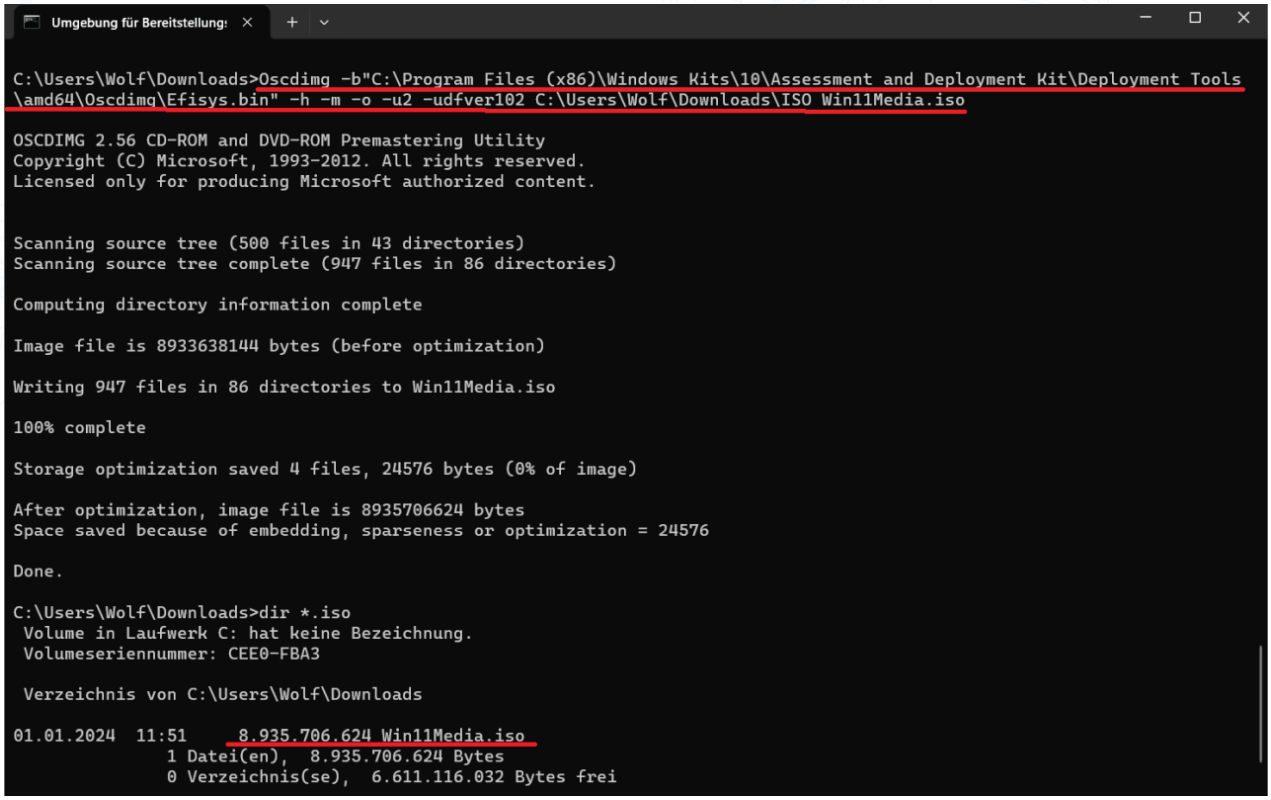
Wenn man für das Erstellen der ISO die Eingabeaufforderung nutzt, für die das ADK eine eigene Verknüpfung einrichtet, dann befindet sich dort oscdimg.exe bereits im Suchpfad.



Die Umgebung für die Bereitstellungstools erweitert den Suchpfad auf oscdimg

Ein Aufruf könnte dann so aussehen:

Oscdimg.exe -b"C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\amd64\Oscdimg\Efisys.bin" -h -m -o -u2 -udfver102 C:\Users\Wolf\Downloads\ISO Win11Media.iso



Bootfähige ISO mit oscdimg.exe erstellen

Erläuterung der Optionen

Mit der Option -b spezifiziert man die erwähnten Dateien für den Boot-Sektor, in unserem Beispiel *Efisys.bin* für das GPT-Layout. Mit -h sorgt man dafür, dass auch versteckte Dateien kopiert werden und -m ignoriert die maximal zulässige Größe für ISO, die durch die Kapazität physischer CDs oder DVDs vorgegeben ist.

Der Schalter -o steht für *optimize* und verhindert mit Hilfe eines MD5-Hash, dass Dateien doppelt kopiert werden. Schließlich legt man mit -u2 fest, dass nur das UDF-Dateisystem angelegt und auf ISO 9660 verzichtet wird. Der Schalter -udfver102 spezifiziert die Version des Dateisystems.

Nach den Optionen folgt als erster Parameter die Wurzel des Verzeichnisbaums, den man in die ISO schreiben möchte. Deren Name folgt als letzter Parameter.

Wenn man alternative Optionen verwenden möchte, dann findet sich eine Erklärung für alle unterstützten Schalter in Microsofts Dokumentation zu Oscdimg. Diese betreffen vor allem die Verwendung anderer Dateisysteme und Multi-Boot-Einträge.

Deployment durch Anwenden eines Images

Verfügt man über ein Custom Image und möchte dieses auf die Zielrechner applizieren, dann muss man sich um einige Aufgaben selbst kümmern, die sonst das Setup übernimmt. Dazu gehören etwa das Partitionieren des Laufwerks, das Hinzufügen der Boot-Dateien oder das Installieren des Recovery Environments (WinRE).

Laufwerk in WinPE mit PowerShell partitionieren

Bevor man ein WIM-Abbild auf einen Rechner übertragen kann, muss man dessen Laufwerk entsprechend partitionieren. Nachdem der ganze Prozess weitgehend automatisiert werden soll, wird man für diese Aufgabe ein Script einsetzen. Dieses kann man zum Beispiel in Windows PE automatisch aus *startnet.cmd* laden.

Microsofts Beispiele dafür beruhen auf Batch-Dateien und Diskpart. Installiert man hingegen, wie oben beschrieben, PowerShell in Windows PE, dann lässt sich diese Aufgabe damit wesentlich eleganter erledigen.

Bei der Bereitstellung von WinPE muss man aber dafür sorgen, dass die Storage-Cmdlets an Bord sind:

```
Dism /Add-Package /Image:mount /PackagePath:"%WinPERoot%\amd64\WinPE_OCs\WinPE-StorageWMI.cab"
```

```
Dism /Add-Package /Image:mount /PackagePath:"%WinPERoot%\amd64\WinPE_OCs\de-de\WinPE-StorageWMI_de-de.cab"
```

Standardmäßig teilt das Setup die erste Disk gemäß Microsofts Empfehlung in drei (BIOS) bzw. vier Partitionen (UEFI) auf. Ein Script kann das gleiche Disk-Layout erzeugen oder ein davon abweichendes realisieren.

Standard-Layout des Systemlaufwerks auf UEFI-PCs

Standardpartitionen per Script einrichten

Das folgende Script sucht ein Laufwerk, das nicht über USB angeschlossen und noch nicht initialisiert ist (PartitionStyle -eq "RAW"). Man kann hier alternativ einfach einen Datenträger anhand seiner Nummer festlegen, zum Beispiel

$Disk = Get-Disk -Number 0

Anschließend wird die Disk bei Bedarf online genommen.

WinPE-PartDisk.ps1

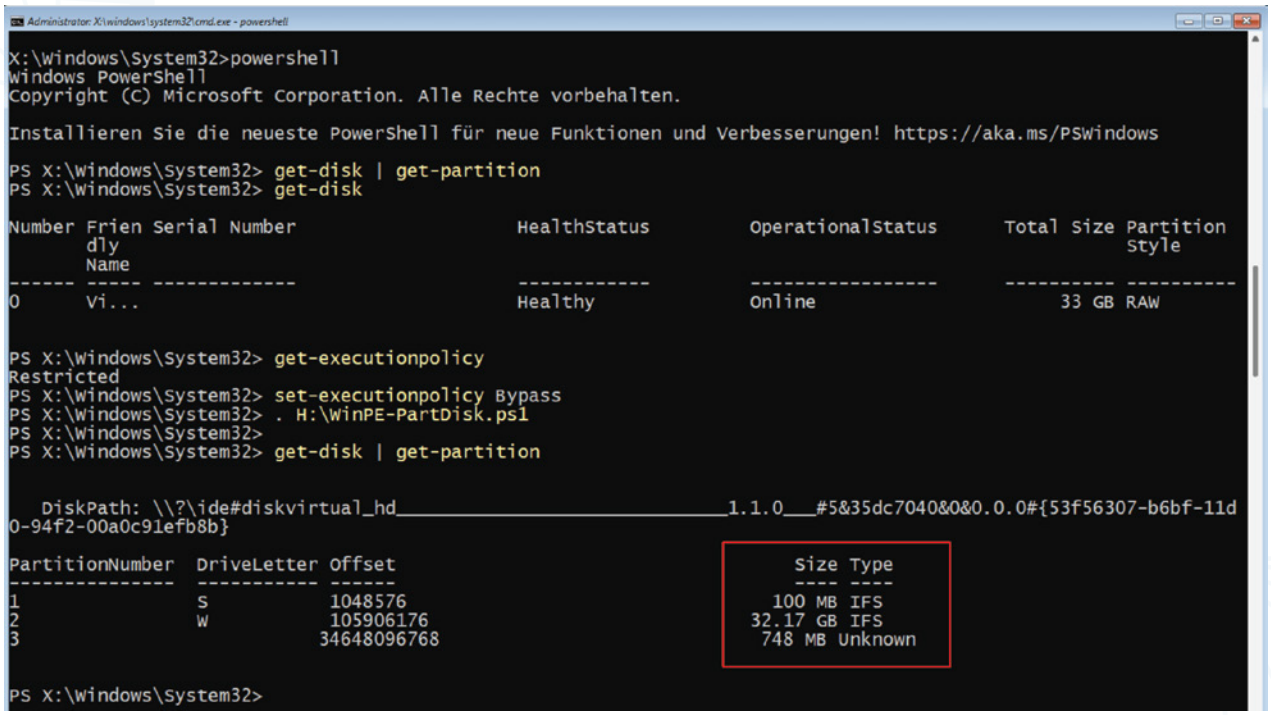
```powershell
$Disk = Get-Disk | where {$_.BusType -NE "USB" -and $_.PartitionStyle -eq "RAW"}

if($Disk.IsOffline){
    $Disk | Set-Disk -IsOffline:$false
    }

if($env:firmware_type -eq "UEFI"){
    #Größe der Windows-Partition berechnen
    $winsize = $disk.Size - 260*1MB - 16*1MB - 750*1MB

    #Disk als GPT initialisieren
    Initialize-Disk -PartitionStyle GPT -Number $Disk.Number

    $EFI = New-Partition -DiskNumber $Disk.Number -Size 260MB -DriveLetter "S" `
    -GptType "{c12a7328-f81f-11d2-ba4b-00a0c93ec93b}" |
    Format-Volume -FileSystem FAT32 -NewFileSystemLabel "System" -Force

    #Microsoft Reserved (MSR)
    $msr = New-Partition -DiskNumber $Disk.Number -Size 16MB `
    -GptType "{e3c9e316-0b5c-4db8-817d-f92df00215ae}"
    }
else{
    #Größe der Windows-Partition berechnen
    $winsize = $disk.Size - 100*1MB - 750*1MB

    #Disk als MBR initialisieren
    Initialize-Disk -PartitionStyle MBR -Number $Disk.Number

    $system = New-Partition -DiskNumber $Disk.Number -Size 100MB -DriveLetter "S" |
    Format-Volume -FileSystem NTFS -NewFileSystemLabel "System" -Force
    }

    #Windows-Partition erstellen
    $win = New-Partition -DiskNumber $Disk.Number -Size $winsize -DriveLetter "W" |
    Format-Volume -FileSystem NTFS -NewFileSystemLabel "Windows" -Force

    #WinRE-Partition erstellen
    $winre = New-Partition -DiskNumber $Disk.Number -UseMaximumSize  -DriveLetter "R" |
    Format-Volume -FileSystem NTFS -NewFileSystemLabel "Recovery" -Force

#WinRE-Partition markieren
if($env:firmware_type -eq "UEFI"){
    Set-Partition -GptType "{DE94BBA4-06D1-4D40-A16A-BFD50179D6AC}" `
    -DriveLetter $winre.DriveLetter
    }
else{
    Set-Partition -MbrType 0x27 -DriveLetter $winre.DriveLetter
    }
```

PowerShell-Script zur Partitionierung des Windows-Laufwerks

Anhand der Umgebungsvariablen $env:firmware kann man feststellen, ob es sich um einen UEFI- oder BIOS-PC handelt. Die Berechnung des Speicherplatzes, der für die Windows-Partition bleibt, ist für die beiden Architekturen verschieden, weil die vom System benötigten Partitionen jeweils andere Größen aufweisen.

Einige Partitionen erhalten einen Laufwerksbuchstaben nach dem Vorbild der Microsoft-Dokumentation für den Fall, dass man anschließend von dort die Batch-Datei zum Aufspielen eines Images verwenden möchte.

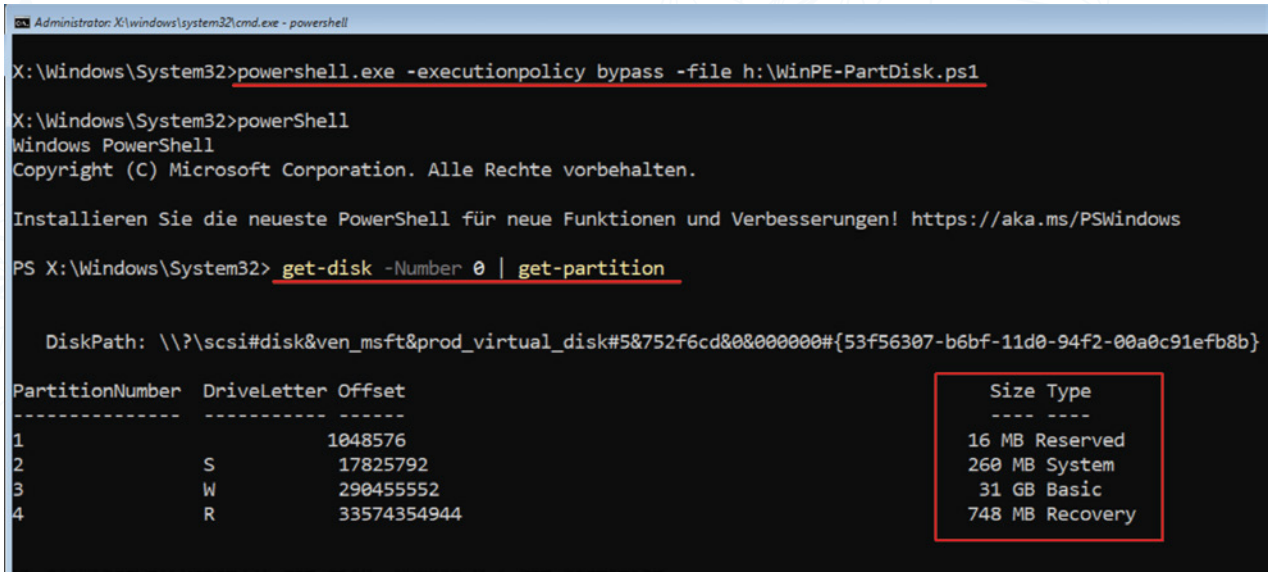
Die Partitionen für EFI und WinRE bekommen eine spezielle Kennzeichnung, im Fall von UEFI einen *GptType* und auf BIOS-PCs einen *MbrType*. Die Partition für WinRE sollte mindestens 250MB groß sein, um später den Fehler 0x80070643 beim Einspielen von Updates zu vermeiden.



Partitionierung eines MBR-Laufwerks auf einem BIOS-Rechner

Wenn man das Script aus WinPE aufrufen möchte, dann sollte man bedenken, dass dort die Execution Policy standardmäßig auf *Restricted* gestellt ist. Dies ändert man entweder interaktiv auf der Kommandozeile oder startet für eine automatische Ausführung eine neue Instanz von PowerShell:

PowerShell -ExecutionPolicy bypass -File .\WinPE-PartDisk.ps1



Erfolgreiche Partitionierung eines GPT-Laufwerks durch das PowerShell-Script

Nach dem erfolgreichen Partitionieren des Laufwerks kann man entweder das Windows-Setup starten (und diesem Windows-Partition über eine autounattand.xml zuweisen) oder ein eigenes WIM mit Hilfe von DISM anwenden.

Code für das Partitionierungs-Script

```
$Disk = Get-Disk | where {$_.BusType -NE "USB" -and $_.PartitionStyle -eq "RAW"}
if($Disk.IsOffline){
   $Disk | Set-Disk -IsOffline:$false
   }
if($env:firmware_type -eq "UEFI"){
   #Größe der Windows-Partition berechnen
   $winsize = $disk.Size - 260*1MB - 16*1MB - 750*1MB

   #Disk als GPT initialisieren
   Initialize-Disk -PartitionStyle GPT -Number $Disk.Number

   $EFI = New-Partition -DiskNumber $Disk.Number -Size 260MB -DriveLetter "S" `
   -GptType "{c12a7328-f81f-11d2-ba4b-00a0c93ec93b}" |
   Format-Volume -FileSystem FAT32 -NewFileSystemLabel "System" -Force
```

```
    #Microsoft Reserved (MSR)
    $msr = New-Partition -DiskNumber $Disk.Number -Size 16MB `
    -GptType "{e3c9e316-0b5c-4db8-817d-f92df00215ae}"
    }
else{
    #Größe der Windows-Partition berechnen
    $winsize = $disk.Size - 100*1MB - 750*1MB

    #Disk als MBR initialisieren
    Initialize-Disk -PartitionStyle MBR -Number $Disk.Number

    $system = New-Partition -DiskNumber $Disk.Number -Size 100MB -DriveLetter "S" |
    Format-Volume -FileSystem NTFS -NewFileSystemLabel "System" -Force
    }
    #Windows-Partition erstellen
    $win = New-Partition -DiskNumber $Disk.Number -Size $winsize -DriveLetter "W" |
    Format-Volume -FileSystem NTFS -NewFileSystemLabel "Windows" -Force

    #WinRE-Partition erstellen
    $winre = New-Partition -DiskNumber $Disk.Number -UseMaximumSize  -DriveLetter "R" |
    Format-Volume -FileSystem NTFS -NewFileSystemLabel "Recovery" -Force

#WinRE-Partition markieren
if($env:firmware_type -eq "UEFI"){
    Set-Partition -GptType "{DE94BBA4-06D1-4D40-A16A-BFD50179D6AC}" `
    -DriveLetter $winre.DriveLetter
    }
else{
    Set-Partition -MbrType 0x27 -DriveLetter $winre.DriveLetter
    }
```

Windows-Image auf Ziel-PCs anwenden

Nachdem man das Systemabbild einer Musterinstallation erfasst hat, dann kann man damit beliebig viele Rechner klonen. Dafür bieten sich mehrere Verfahren an. Ein gängiges Verfahren besteht darin, die Ziel-PCs mit WinPE von USB oder via PXE zu booten und anschließend dort das Image mit DISM oder PowerShell aufzuspielen.

Dabei muss man sich um einige Aufgaben, um die sich sonst das Setup-Programm erledigt, jedoch selbst kümmern. Dazu gehören das Bereitstellen von Windows PE, das Partitionieren des Windows-Laufwerks, das Kopieren der Boot-Dateien und das Einrichten des Recovery Environments.

WinPE kann man von einem externen Datenträger starten oder über PXE laden, wobei es für das weitere Vorgehen von Vorteil ist, wenn man die PowerShell-Pakete installiert hat. In diesem Fall kann man statt der alten CLI-Tools die moderneren Cmdlets verwenden.

Für eine weitgehende Automatisierung der Installation ist es sinnvoll, die hier folgenden Befehle in einem Script zusammenzufassen.

Golden Image aufspielen

Sobald man einen Rechner, auf dem Windows 11 installiert werden soll, von WinPE gebootet hat, richtet man die Partitionen mit dem oben vorgestellten Script ein oder verwendet dafür die von Microsoft bereitgestellten Batch-Dateien.

Anschließend verbindet man in Windows PE das Netzlaufwerk, auf dem das Image abgelegt ist. Mit PowerShell funktioniert das nach diesem Muster:

```
New-SmbMapping -LocalPath P: -RemotePath "\\server\pfad" `

-UserName "contoso\me" -Password "P@ssw0rd"
```

Alternativ kann man das alte *net use* verwenden:

```
net use p: \\server\verzeichnis /user:contoso\user
```

Dieser Schritt entfällt natürlich, wenn man die WIM-Datei auf einem lokalen Datenträger mitführt.

Anschließend muss sichergestellt sein, dass sich die Partitionen für Windows, das System und WinRE über Laufwerksbuchstaben ansprechen lassen, im Folgenden sind dies W:, S: und R:. Das oben erwähnte PowerShell-Script sowie Microsofts Batch-Dateien ordnen diese Buchstaben automatisch zu.

Nun kann man das Image auf dem Rechner anwenden. Mit DISM sieht der entsprechende Befehl so aus, wobei in unserem Beispiel das WIM-Archiv auf P: liegt:

```
dism.exe /Apply-Image /ImageFile:P:\MyImage.wim /Index:1 /ApplyDir:W:\
```

Alternativ erledigt ein äquivalenter Aufruf von PowerShell diese Aufgabe:

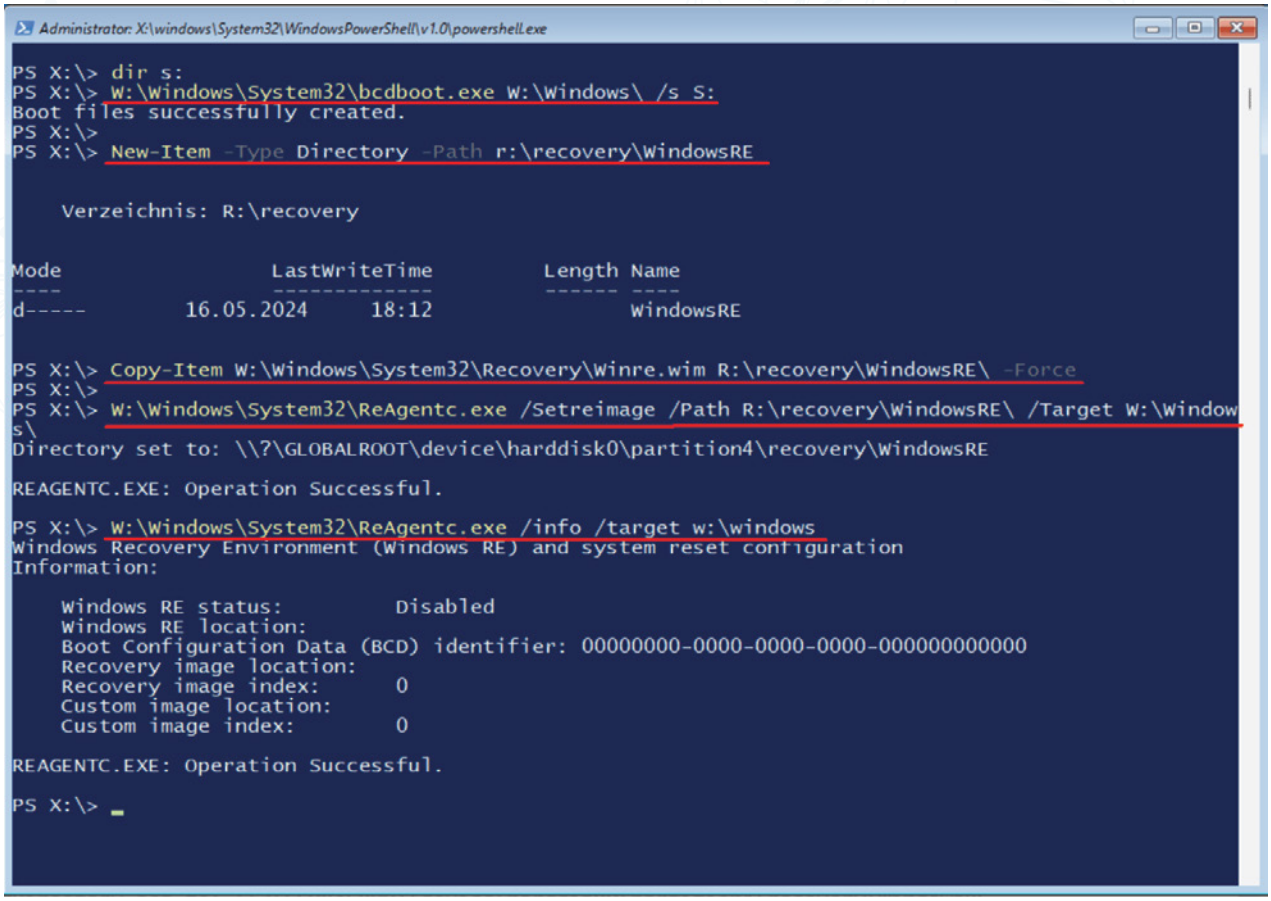
```
Expand-WindowsImage -ImagePath "P:\MyImage.wim" -ApplyPath "w:\" -Index 1
```

Der Parameter *Index* muss immer angegeben werden, auch wenn das WIM-Archiv nur ein Image enthält.

Boot-Dateien kopieren

Nach erfolgreicher Übertragung des Images auf den Rechner kopiert man die Boot-Dateien in die Systempartition:

```
W:\Windows\System32\bcdboot.exe W:\Windows /s S:
```



Boot-Dateien in die Systempartition kopieren und WinRE einrichten

Windows RE installieren

Abschließend richtet man WinRE ein:

md R:\Recovery\WindowsRE

xcopy /h W:\Windows\System32\Recovery\Winre.wim R:\Recovery\WindowsRE\

W:\Windows\System32\Reagentc /Setreimage /Path R:\Recovery\WindowsRE /Target W:\Windows

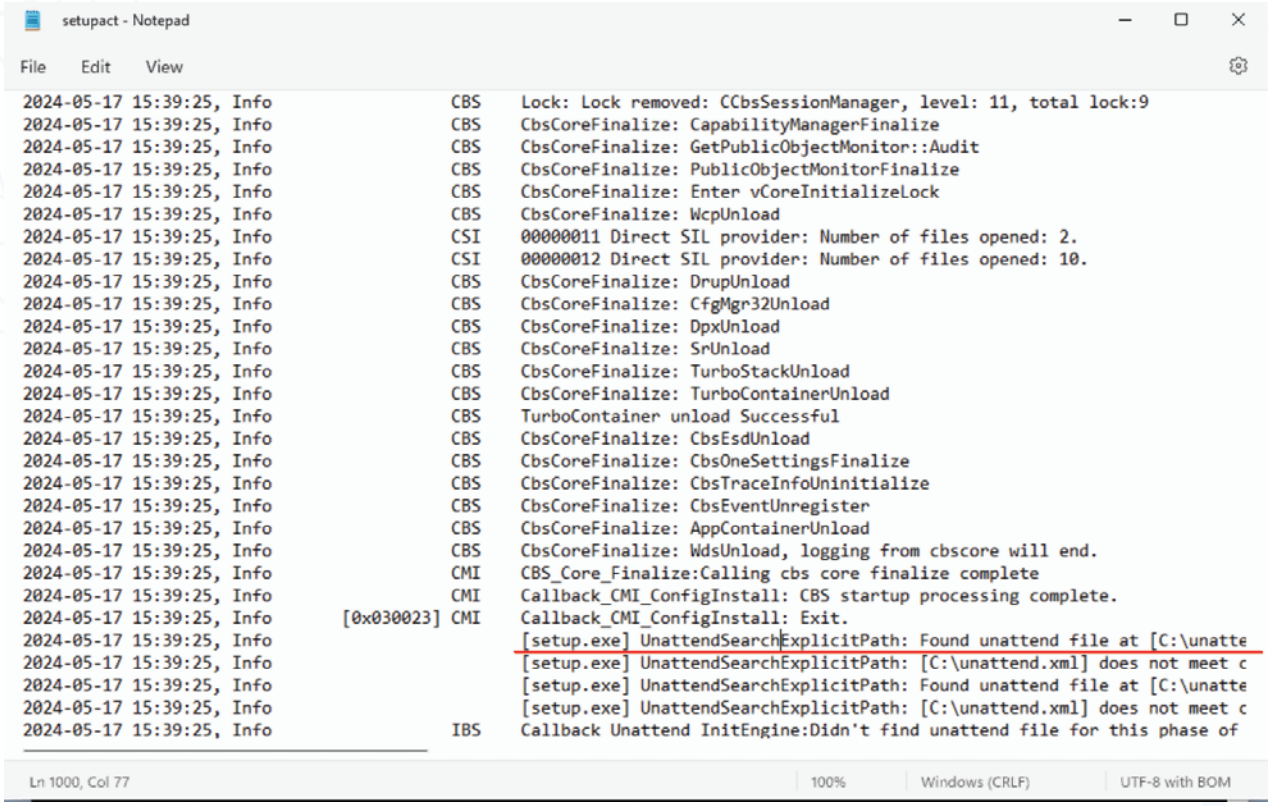
Windows-Installation abschließen

Im letzten Schritt startet man den Rechner neu, etwa durch Eingabe von

Restart-Computer

Im Anschluss daran fährt das System vom frisch installierten Windows hoch, richtet die Hardware ein und sucht nach Updates. Der Vorgang mündet dann in die OOBE-Phase mit ihren zahlreichen Dialogen zur Konfiguration von Windows.

Diese kann man den Benutzern ersparen, indem man eine Antwortdatei im Image hinterlegt. Als Speicherort für die unattend.xml kommen offiziell verschiedene Verzeichnisse in Frage, bei meinem Test wurde sie aber nur im Wurzelverzeichnis abgearbeitet.



In der Datei \windows\panther\setupact.log kann man prüfen, ob die Antwortdatei abgearbeitet wurde

Am besten kopiert man sie schon online in der Musterinstallation dorthin, andernfalls kann man das durch Mounten des Offline-Images nachholen.

Windows 11 mit OSDCloud installieren

OSDCloud ist ein freies PowerShell-Framework für das Windows-Deployment. Nach dem Booten von einem angepassten WinPE installiert man Windows über die OSDCloudGUI oder per Script. In beiden Fällen kommt *Expand-WindowsImage* zum Einsatz, um die WIM-Datei auf den Ziel-PC anzuwenden.

Ziel der Entwickler um David Segura war es, eine Deployment-Lösung für Windows zu erschaffen, die den Vorgang weitgehend automatisieren und ohne lokale Infrastruktur auskommen kann. Interessant sind die unkomplizierten Möglichkeiten, mit denen man ein Image um Treiber oder andere Software ergänzen und dessen Einstellungen konfigurieren kann.

Standardmäßig lädt OSDCloud die Installationsmedien von Microsoft herunter, nachdem der Rechner von einem angepassten WinPE gebootet hat und das Setup-Script ausführt. Man kann allerdings auch ein Windows Custom Image verwenden und dieses von einem eigenen Online-Storage oder von einem lokalen Verzeichnis laden.

Nachdem OSDCloud die Deployment-Tools aus dem ADK voraussetzt, müssen diese ebenso installiert sein wie die Dateien für Windows PE, die man seit geraumer Zeit separat herunterladen muss.

OSDCloud installieren

Im ersten Schritt fügt man das OSD-Modul hinzu:

```
Install-Module OSD -Force
```

Anschließend kann man sich vom erheblichen Funktionsumfang des OSD-Moduls überzeugen:

```
Get-Command -Module OSD
```

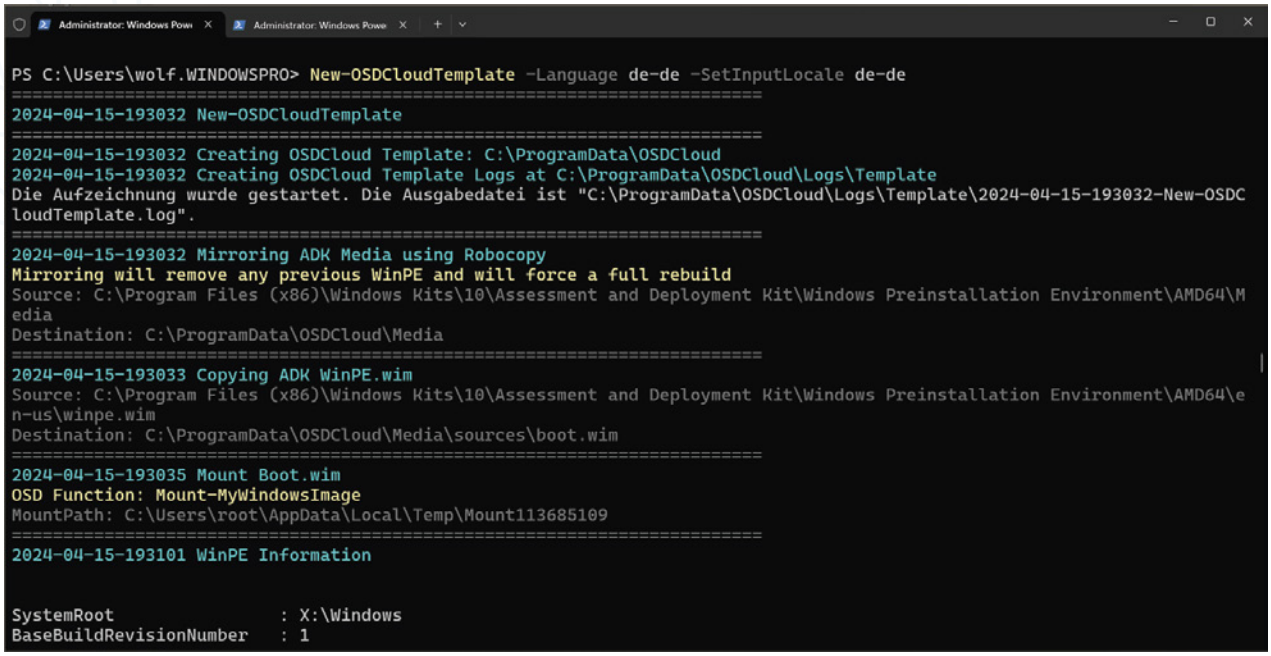
Template für WinPE erzeugen

OSD operiert mit Templates, mit denen man anschließend mehrere Workspaces befüllen kann, aus denen dann letztlich die Medien generiert werden. Auf diese Weise kann man unterschiedlich konfigurierte Images aus einer Vorlage erzeugen.

OSDCloud nutzt ein speziell konfiguriertes Windows PE, von dem man einen Rechner bootet und das anschließend das OS-Deployment startet. Daher erzeugt man zuerst ein Template für WinPE, in diesem Fall mit deutscher Lokalisierung:

New-OSDCloudTemplate -Language de-de -SetInputLocale de-de

Diese Operation dauert einige Zeit und durchläuft zahlreiche Schritte, wobei alle benötigten Dateien aus dem Quellordner kopiert werden. Ein Zielverzeichnis gibt man dafür nicht an, OSDCloud nimmt dafür automatisch *$env:ProgramData\OSDCloud*.

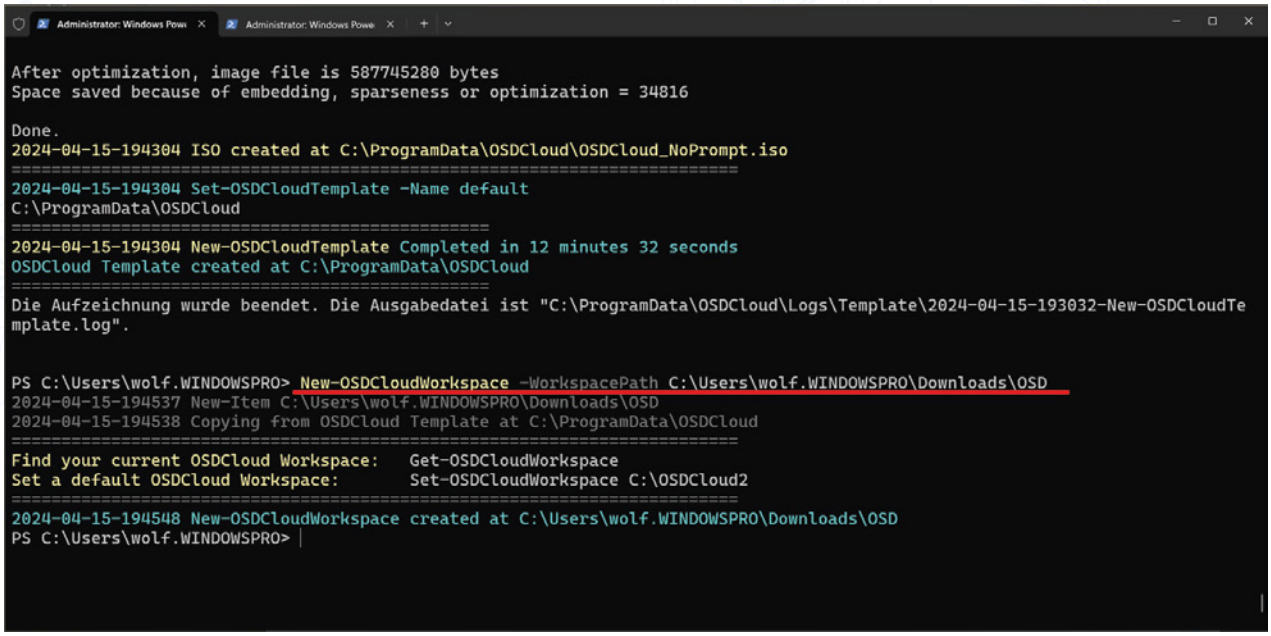


Neue Vorlage für das Erzeugen von WinPE-Images erstellen

Workspace für OSD anlegen

Auf Basis dieses Templates erzeugt man nun einen neuen Workspace:

New-OSDCloudWorkspace -WorkspacePath <LW:\Pfad\MyWorkspace>



Workspace für ein WinPE-Image erstellen

Grundsätzlich könnte man nun bereits ein WinPE-Boot-Medium erstellen. Dieses Universal WinPE enthält bereits zahlreiche Komponenten, die man sonst einzeln von Hand hinzufügen muss.

Dazu gehören zahlreiche Packages für .NET und PowerShell, Tools wie curl und setx sowie der Fix *wgl4_boot.ttf* gegen Darstellungsprobleme. OSDCloud konfiguriert zudem die Execution Policy für PowerShell und fügt die Unterstützung für die PowerShell Gallery hinzu.

Eine wesentliche Fähigkeit von OSDCloud besteht indes darin, das Deployment zu automatisieren und dabei etwa Treiberpakete dynamisch nachzuladen, WLAN-Profile zu importieren oder Scripts auszuführen. Zu diesem Zweck passt man das WinPE-Image noch weiter an. Dabei kann man auch den automatischen Start der Windows-Installation veranlassen.

WinPE-Image anpassen

Diesem Zweck dient die Funktion *Edit-OSDCloudWinPE*. In der Praxis sollte man dieses Cmdlet nur einmal und dabei gleich mit allen erforderlichen Parametern aufrufen, weil es bei jeder Ausführung das Image mounten, bearbeiten und wieder aushängen muss. Außerdem überschreibt es jedes Mal die zuvor gemachten Änderungen.

Will man einen anderen als den aktuellen Workspace bearbeiten, dann muss man dessen Pfad über den Parameter WorkspacePath spezifizieren. Mit

```
Get-OSDCloudWorkspace
```

findet man heraus, welcher gerade im Zugriff ist.

Treiber hinzufügen

Um zum Beispiel die Treiber für einen Dell-PC automatisch von der Website des Herstellers herunterzuladen, würde man sie so aufrufen:

```
Edit-OSDCloudWinPE -CloudDriver Dell
```

Bei Bedarf kann man auch mehrere Quellen angeben, etwa "Dell,Intel", oder sämtliche mit dem Wildcard '*'. Sind Treiber bereits lokal abgelegt, dann gibt man den Pfad dorthin mit dem Parameter *DriverPath* an.

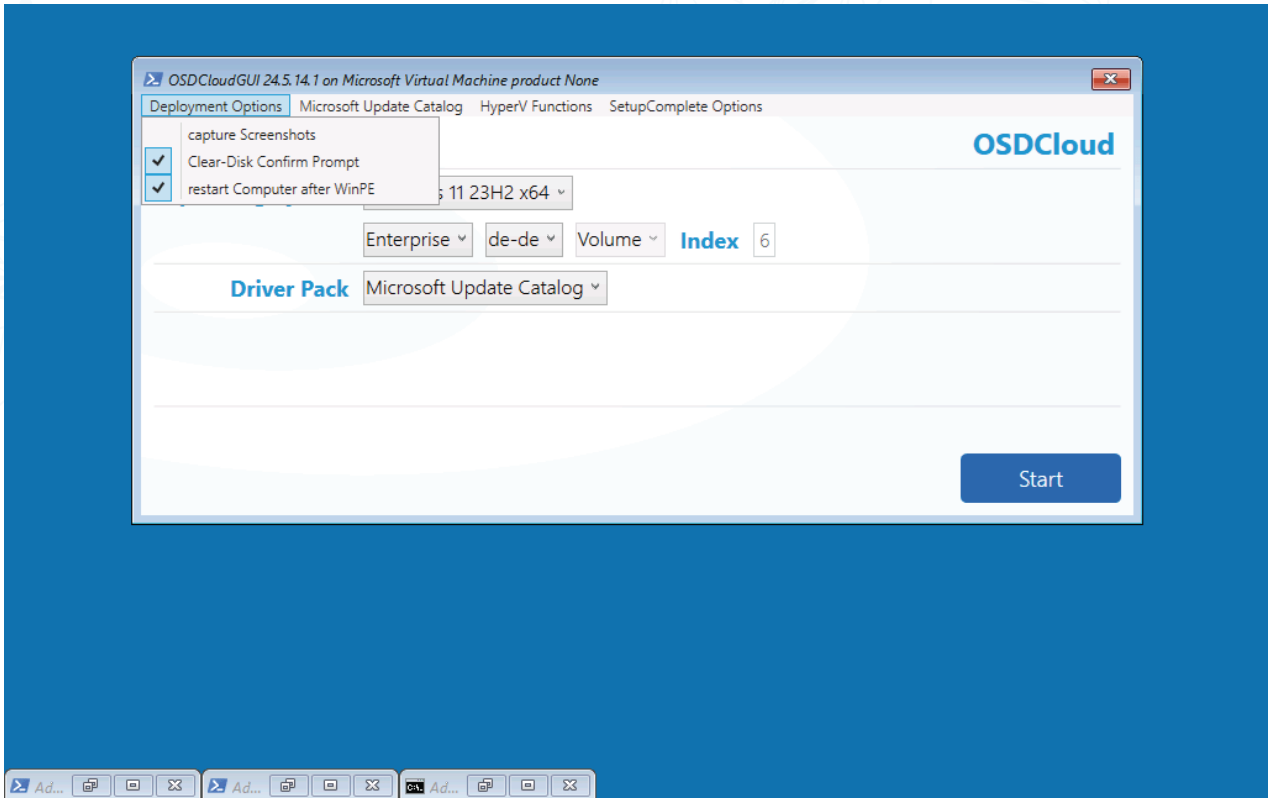
Mit dem Parameter *PSModuleCopy* übernimmt man PowerShell-Module vom aktuellen Betriebssystem, mit *PSModuleInstall* von der Gallery.

Installation über GUI oder Script vorgeben

OSDCloud bietet sowohl eine grafische Oberfläche, über die man in WinPE die Installation anpassen kann, als auch ein rein Script-basiertes Verfahren, das ohne Intervention des Benutzers auskommt.

Soll nach dem Booten von WinPE die OSDCloud-GUI automatisch starten, dann passt man das Image so an:

```
Edit-OSDCloudWinPE -StartOSDCloudGUI
```



Grafische Oberfläche zur Auswahl der Setup-Parameter

Bevorzugt man hingegen eine Zero-Touch-Installation, dann kann man den Aufruf von *Start-OSDCloud* in die startnet.cmd schreiben lassen:

Edit-OSDCloudWinPE -StartOSDCloud "-OSVersion 'Windows 11' -OSBuild 23H2 -OSEdition Enterprise -OSLanguage de-de -OSActivation Volume -ZTI -Restart"

Die Parameter entsprechen den Optionen, die man auf der GUI auswählen kann. Der Schalter ZTI unterdrückt Rückfragen vor dem Löschen des Laufwerks.

Möchte man noch weitere Programme automatisch ausführen, dann kann man diese folgendermaßen in die Startdatei einfügen:

Edit-OSDCloudWinPE -Startnet "Befehl für die Startnet.cmd"

Das entsprechende Kommando wird aber dann nach dem Aufruf von *Start-OSDCloud* ausgeführt, so dass etwa die Zuordnung eines Netzlaufwerks, auf dem sich die Images befinden, zu spät kommt. Soll ein Befehl vor *Start-OSDCloud* zum Zug kommen, dann muss man statt *Startnet* den Parameter *StartPSCommand* verwenden.

Eine vollständige Liste der Parameter findet sich in der Dokumentation.

Angepasstes Image mit OSDCloud verwenden

Die beiden oben vorgestellten Methoden ziehen die Installationsdateien von Microsofts Website und verwenden somit die dort enthaltene install.wim. Viele Admins bevorzugen hingegen ein Custom Image, das sie von einer Musterinstallation erfasst haben.

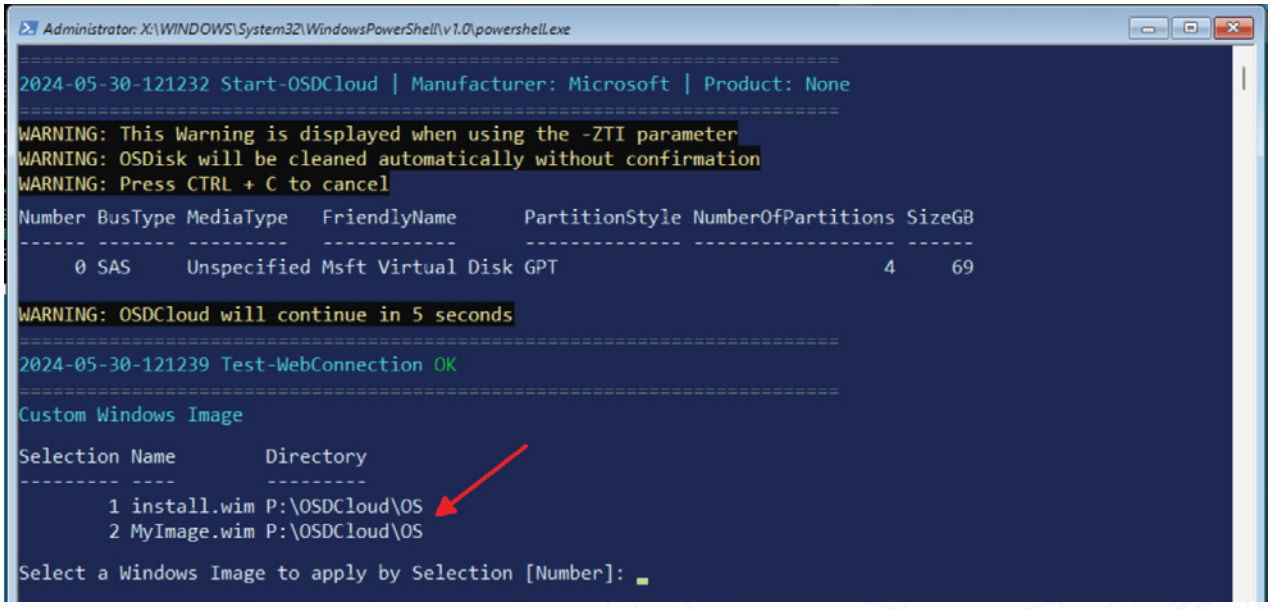
OSDCloud sieht mehrere Optionen für den Ablageort eines solchen Images vor. Dieser kann ein Netzlaufwerk, ein Web-Server oder ein Azure Storage Account sein. Den Speicherort teilt man *Start-OSDCloud* über die Parameter *ImageFileUrl* oder *FindImageFile* mit.

Ersterer erwartet offensichtlich eine URL, während *FindImageFile* dazu führt, dass *Start-OSDCloud* auf allen Laufwerken unter \OSDCloud\OS\ und seinen Unterverzeichnissen nach WIM-Archiven sucht. Verwendet man eine Netzfreigabe, dann muss man dieser also zuerst einen Laufwerksbuchstaben zuordnen.

Um die startnet.cmd im WinPE-Image entsprechend anzupassen, könnte man einen Befehl nach diesem Muster nutzen:

Edit-OSDCloudWinPE -StartPSCommand "net use p: \\192.168.0.33\images /user:contoso\admin P@ssword" -StartOSDCloud "-zti -FindImageFile -Restart"

Wenn OSDCloud dann unter P:\OSDCloud\OS\ Images findet, dann zeigt es diese in einem Menü an, aus dem man das gewünschte auswählen kann.

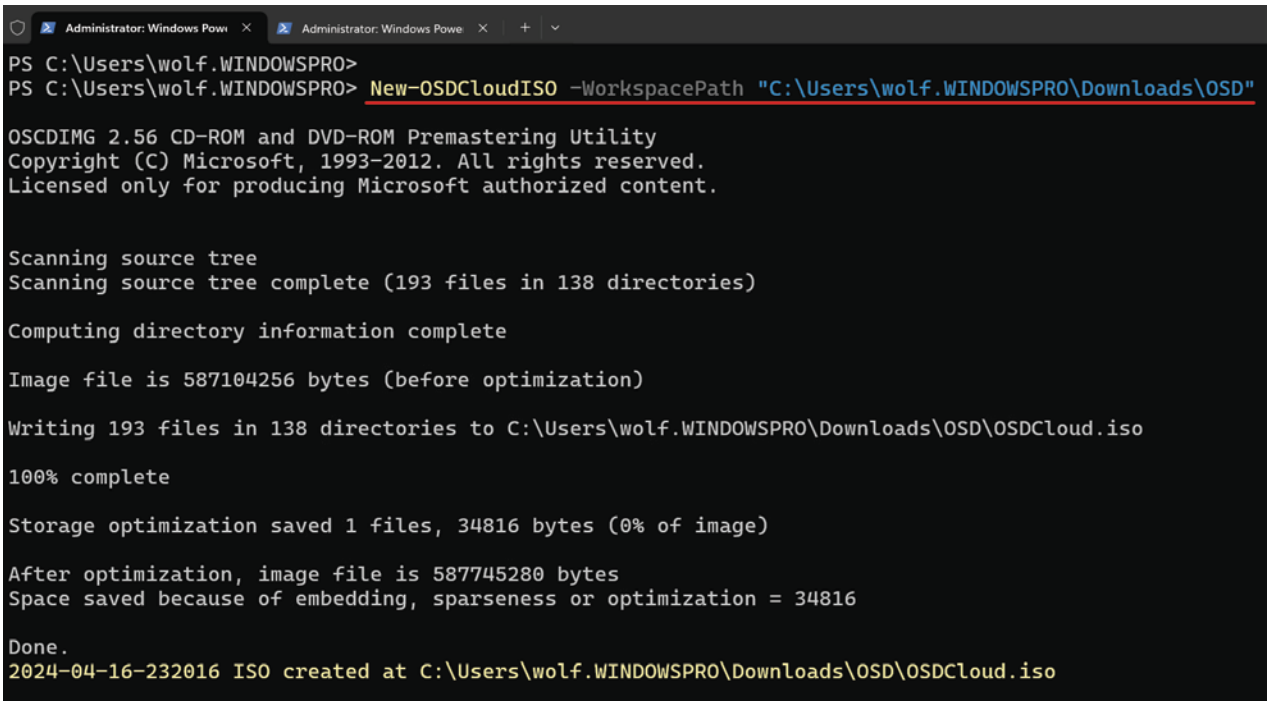


OSDCloud zeigt ein Menü zur Auswahl des Systemabbilds, wenn es WIMs auf Laufwerken unter \OSDCloud\OS\ findet

Datenträger erstellen

Zum Abschluss schreibt man das WinPE-Image in eine ISO oder auf einen USB-Stick. Zuständig dafür sind die Funktionen New-OSDCloudISO bzw. New-OSDCloudUSB.

Wenn man sie ohne Argumente aufruft, dann operieren sie auf Basis des aktuellen Workspace. Alternativ kann man mit *fromIsoFile* bzw. *fromIsoUrl* eine bestehende ISO als Quelle angeben.

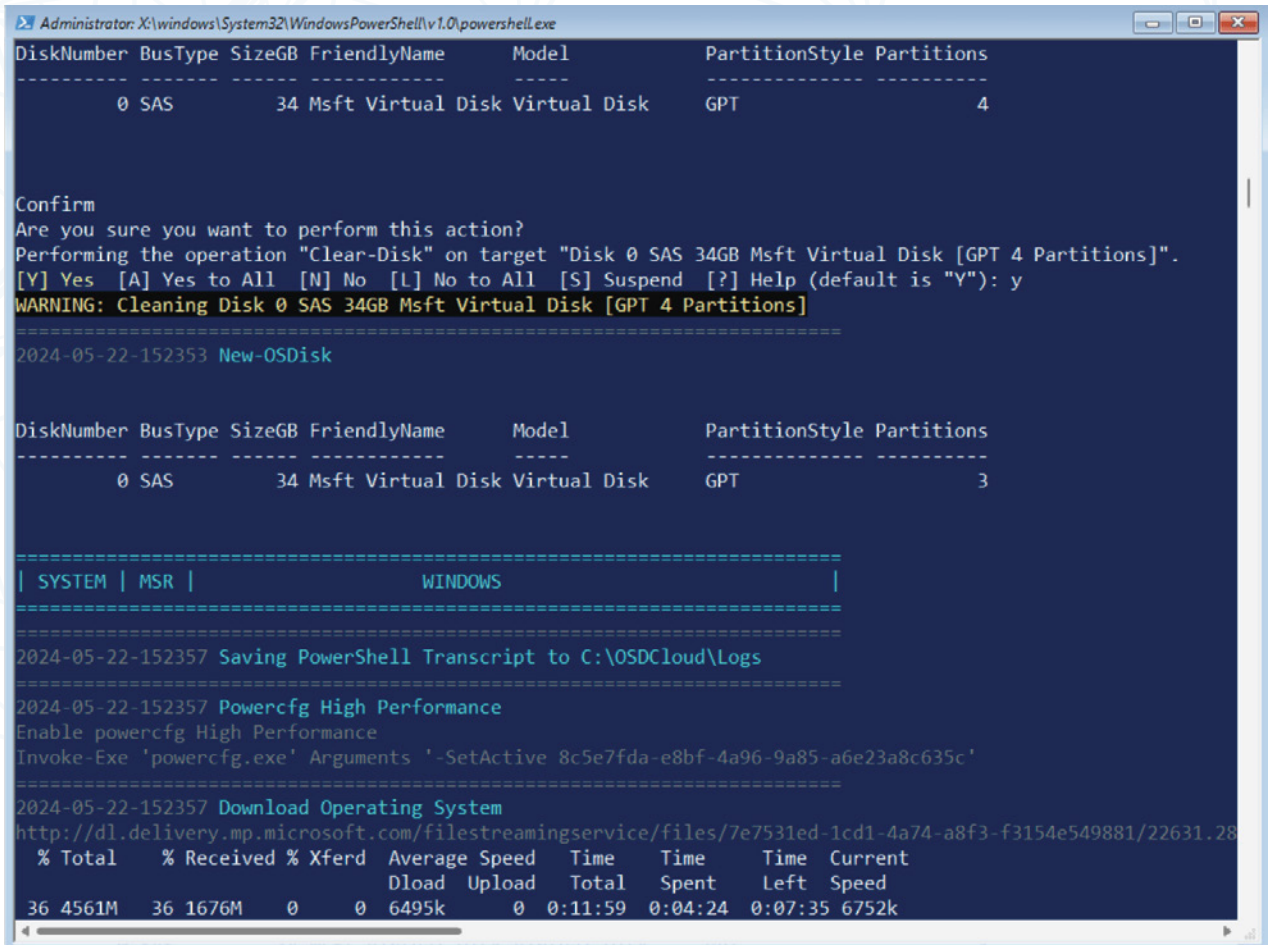


OSDCloud ruft für das Erzeugen der ISO oder des USB-Stick OSCDIMG aus dem ADK auf

New-OSDCloudUSB listet die verfügbaren USB-Laufwerke auf und bietet ein Menü, um den gewünschten Datenträger auszuwählen. Erzeugt man eine ISO, dann landet diese im Workspace, und zwar gleich in doppelter Ausführung. Jene mit dem Namen *OSDCloud_NoPrompt.iso* bootet, ohne dass man eine Taste betätigen muss.

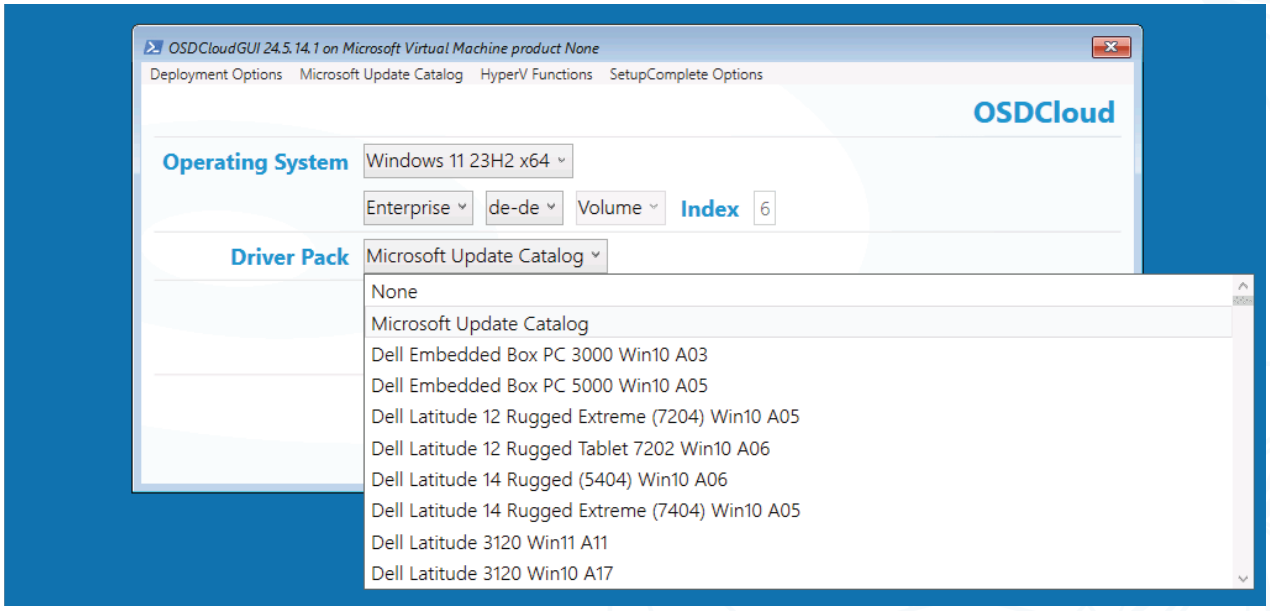
Installation von Windows anstoßen

Bootet man einen Rechner von dem erstellten WinPE-Medium, dann zeigt er abhängig von der oben gewählten Methode eine GUI oder startet automatisch das Setup-Script. Die Optionen auf der grafischen Oberfläche sind weitgehend selbsterklärend.



Installation von Windows 11 Enterprise mit OSDCloud

Interessant ist dabei die Möglichkeit, Treiberpakete zu laden oder Screenshots von der Installation für die spätere Dokumentation anfertigen zu lassen. Diese speichert das Script unter C:\OSDCloud\ScreenPNG.



Auf der OSDCloudGUI kann man Treiber für zahlreiche PC-Modelle verschiedener Hersteller auswählen

In beiden Fällen landen die Benutzer nach Abschluss der Installation bei den OOBE-Dialogen. OSDCloud bietet zur Automatisierung von Aktionen in dieser Phase ein eigenes Cmdlet namens Start-OOBEDeploy. Dieses überspringt die ganzen Dialoge für die Einstellungen zur Privatsphäre aber nicht.

Verwendet man ein angepasstes Windows-Image, dann kann man dort für diesen Zweck die oben beschriebene Antwortdatei hinterlegen.

Installation durch Automatisierung des Setup

Neben dem Anwenden eines Custom Image besteht nach wie vor eine gängige Methode für die unbeaufsichtigte Installation von Windows darin, das Setup mit einer Antwortdatei zu automatisieren. Diese weist allen Einstellungen, die man bei der interaktiven Installation über zahlreiche Dialoge konfiguriert, Werte zu, ohne dass der Benutzer eingreifen muss.

Dieses Verfahren beschränkt sich nicht auf die Verwendung der standardmäßigen install.wim, vielmehr kann man auch hier ein Custom Image verwenden.

Setup mit autounattend.xml steuern

Das Setup-Programm sucht automatisch eine Datei namens autounattend.xml im Wurzelverzeichnis von Windows PE. Man kann aber stattdessen eine Datei beliebigen Namens direkt beim Aufruf von setup.exe angeben:

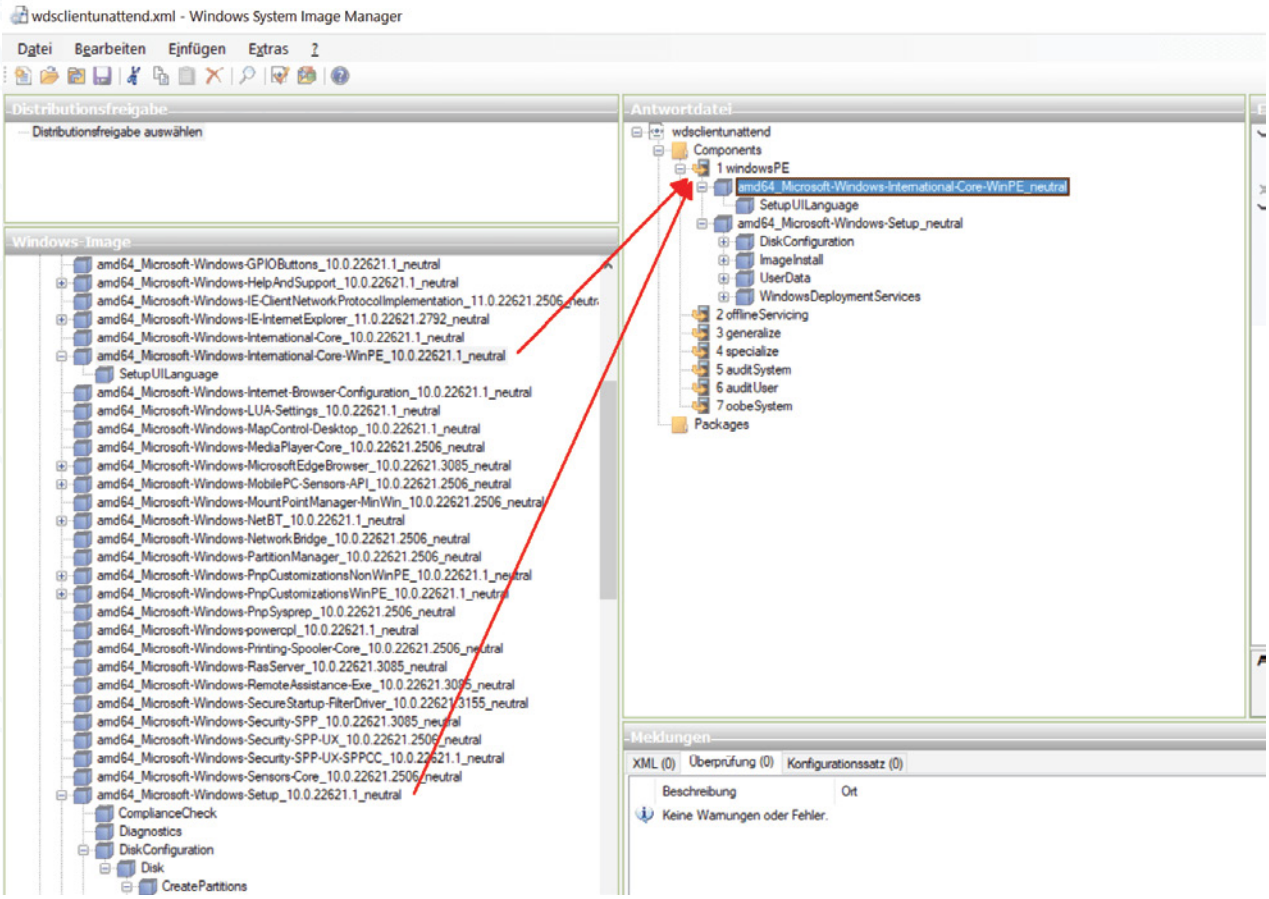
```
setup /unattend:\\server\share\Antwortdatei.xml
```

Entscheidet man sich für die Verwendung einer autounattend.xml, dann muss man diese beim Erstellen des WinPE-Datenträgers in die boot.wim kopieren.

Einstellungen für Antwortdatei konfigurieren

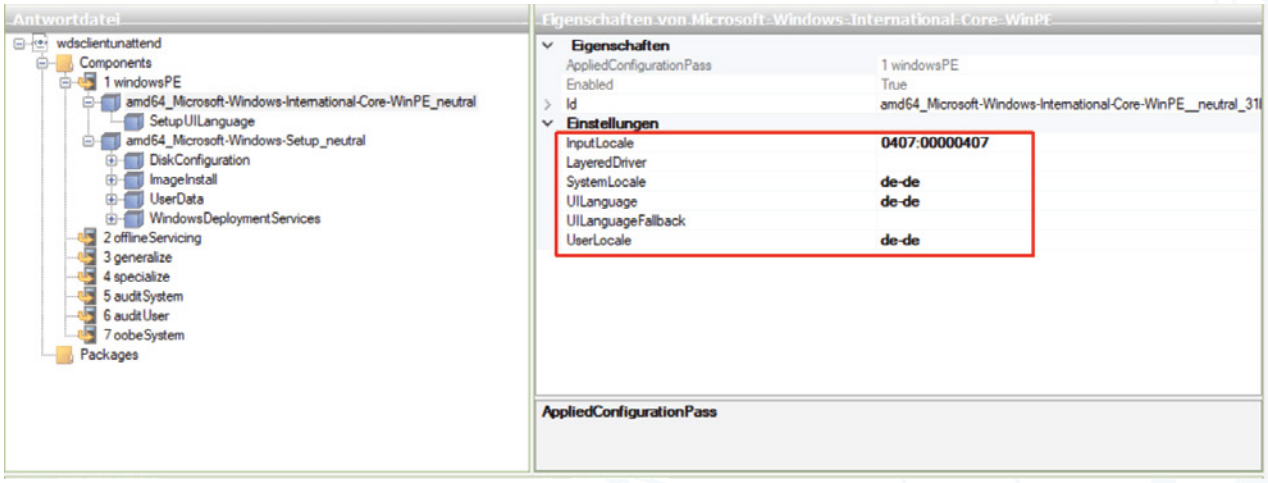
Die autounattend.xml sollte erwartungsgemäß solche Einstellungen enthalten, die in der Installationsphase *WindowsPE* abgearbeitet werden. Dazu gehören jene, die man beim interaktiven Setup im Dialog zur Auswahl von Sprache und Tastatur auswählt, sowie die anschließende Partitionierung des Systemlaufwerks.

Im ersten Schritt erstellt man im Windows System Image Manager (SIM) eine neue Antwortdatei und zieht anschließend die Komponenten *amd64_Microsoft-Windows-International-Core-WinPE__neutral* und *amd64_Microsoft-Windows-Setup__neutral* auf den Knoten *windowsPE* im Fenster *Antwortdatei*.



Komponenten zur Konfiguration der Antwortdatei auf den Abschnitt windowsPE ziehen

Um den Dialog für die regionalen Einstellungen zu überspringen, gibt man unter *amd64_Microsoft-Windows-International-Core-WinPE__neutral* den gewünschten Ländercode bzw. den Wert für das Tastatur-Layout ein. Für Deutschland wäre dieser 0407:00000407.



Regionale Einstellungen über die Antwortdatei festlegen

Zusätzlich kann man bei Bedarf noch unter *SetupUILanguage* die Anzeigesprache für das Setup auswählen.

Disk konfigurieren

Etwas aufwändiger gestaltet sich die Partitionierung des Windows-Laufwerks. Eine Antwortdatei erweist sich dabei als relativ unflexibel, weil man nur die letzte Partition dynamisch erweitern kann. Beim normalen Layout handelt es sich dabei um WinRE.

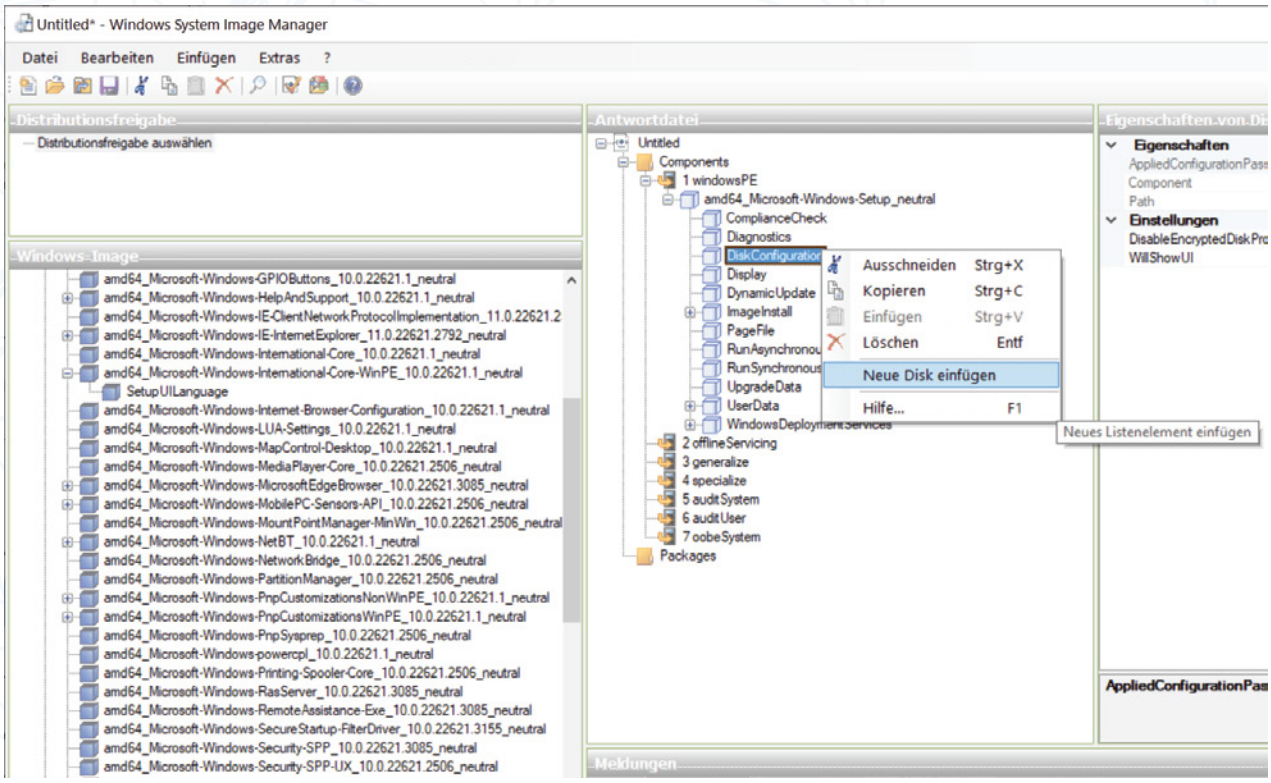
Allerdings möchte man in der Regel nach dem Einrichten der drei anderen Partitionen den ganzen verbleibenden Speicherplatz für Windows reklamieren, was aber nicht klappt, wenn sich dessen Partition an Position 3 befindet.

Standardmäßiges Disk-Layout auf einem UEFI-Rechner

Man kann sich dadurch behelfen, dass man die Windows-Partition ganz nach hinten verlagert.

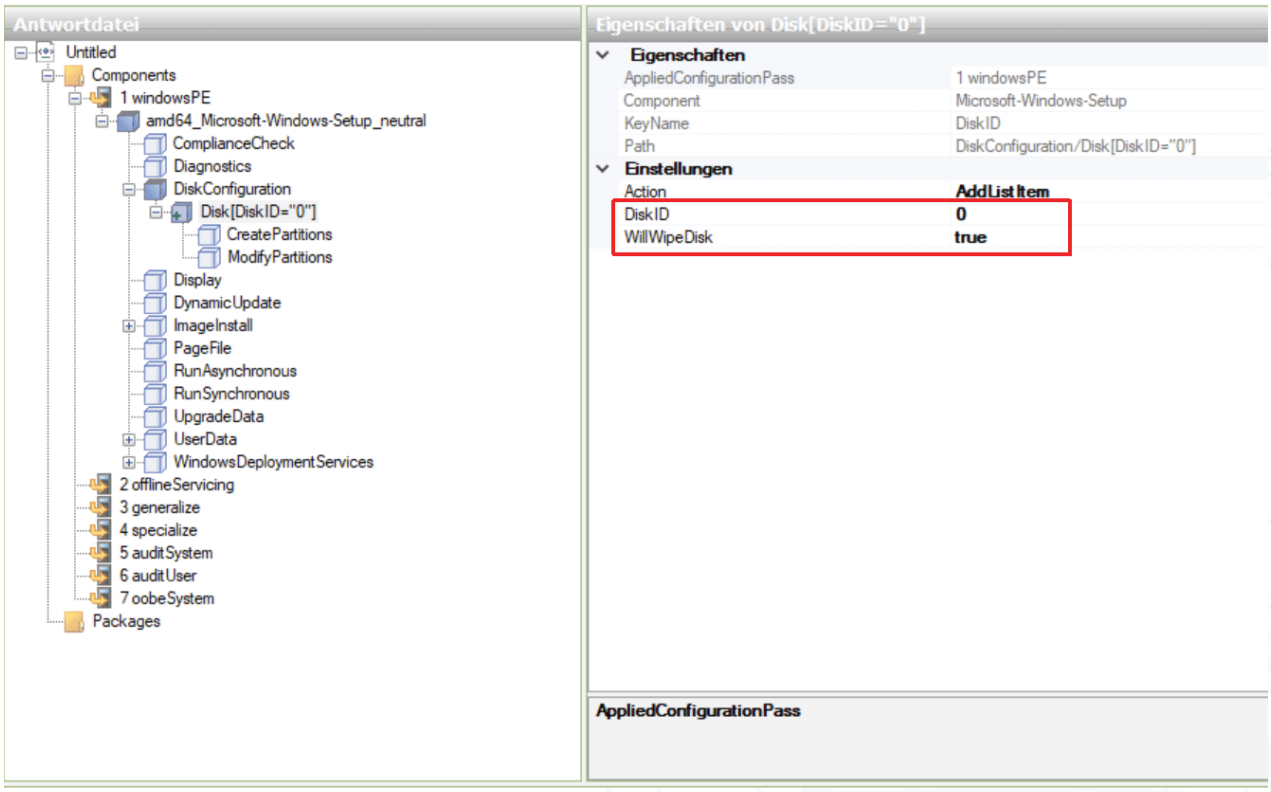
Alternativ berechnet man die verbleibende Größe nach Abzug des Speicherplatzes für die drei anderen Partitionen. Dies ist ein gangbarer Weg, wenn alle Rechner die gleiche Speicherkapazität aufweisen. Andernfalls muss man die Partitionen nachträglich per Script anpassen, damit kein Speicherplatz ungenutzt bleibt.

Für die automatische Partitionierung mittels Antwortdatei öffnet man das Kontextmenü von *DiskConfiguration* und führt dort den Befehl *Neue Disk einfügen* aus.



Laufwerk für die Partitionierung hinzufügen

In den Einstellungen der neuen Disk legt man die DiskID fest, in der Regel ist diese 0. *WillWipeDisk* setzt man auf *true*, um alle vorhandenen Partitionen zu löschen.

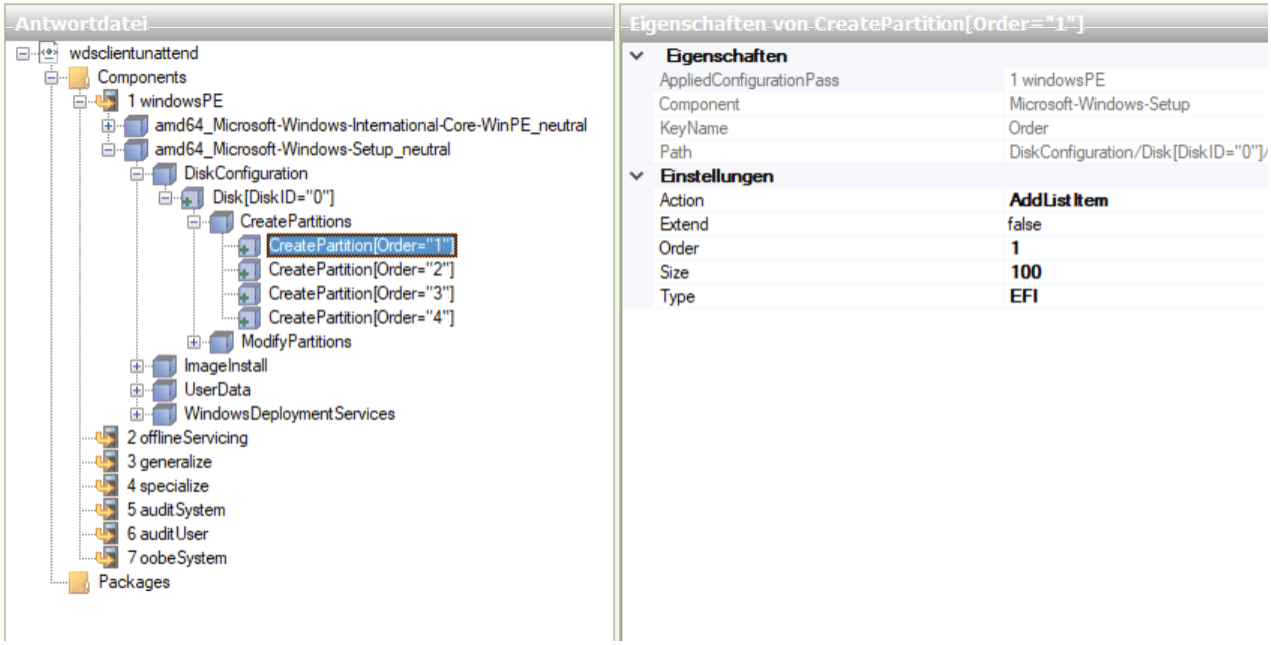


Einstellungen für die neue Disk anpassen

Partitionen hinzufügen

Im nächsten Schritt führt man aus dem Kontextmenü von *CreatePartitions* den Befehl *Neue CreatePartition einfügen* aus. Hier gibt man die Werte für die EFI-Partition ein:

- Extend: false
- Order: 1
- Size: 100
- Type: EFI



Einstellungen für die EFI-Partition

Diesen Vorgang wiederholt man für die drei verbleibenden Partitionen, wobei man folgende Werte wählt:

MSR-Partition

- Extend: false
- Order: 2
- Size: 16
- Type: MSR

Windows-Partition

- Extend: false
- Order: 3
- Size: 65000 (diesen Wert für die Kapazität der jeweiligen Disk berechnen)
- Type: Primary

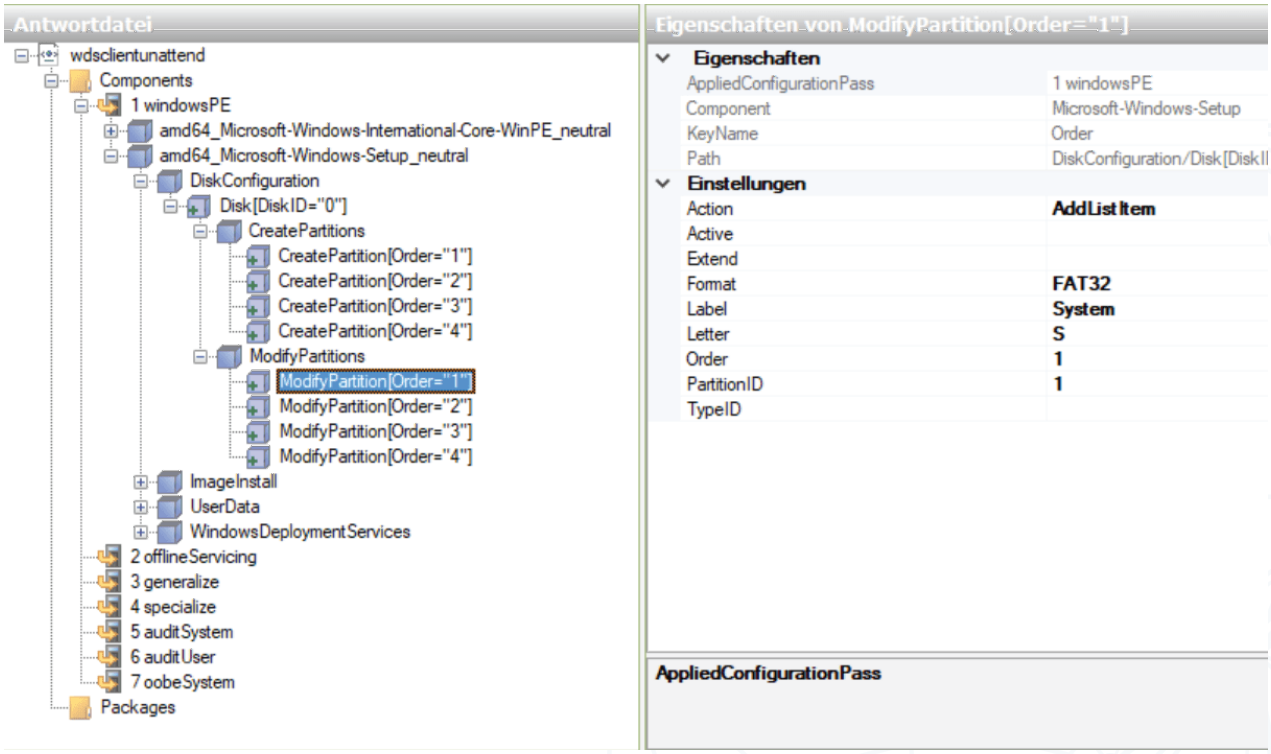
WinRE-Partition

- Extend: true
- Order: 4
- Type: Primary

Partitionen anpassen

Nach dem Anlegen der Partitionen muss man diese noch konfigurieren. Für diese Aufgabe öffnet man das Kontextmenü von *ModifyPartitions* und führt *Neue ModifyPartition einfügen* aus.

Durch die Eingabe bei *Order* bezieht man sich auf die Partition, die man mit der gleichen Nummer zuvor angelegt hat. Der Wert 1 steht für die EFI-Partition. Als *PartitionID* vergibt man am besten die gleiche Kennung wie für *Order*, also ebenfalls die 1. Als Dateisystem muss man hier FAT32 nehmen.



Anpassungen für die EFI-Partition

Auch diesen Vorgang muss man für alle Partitionen ausführen. Dabei braucht man für die MSR-Partition kein Dateisystem und für die beiden anderen NTFS. Beispieleinstellungen kann man dem folgenden Ausschnitt aus der Antwortdatei entnehmen, wobei Laufwerksbuchstaben und Labels optional sind.

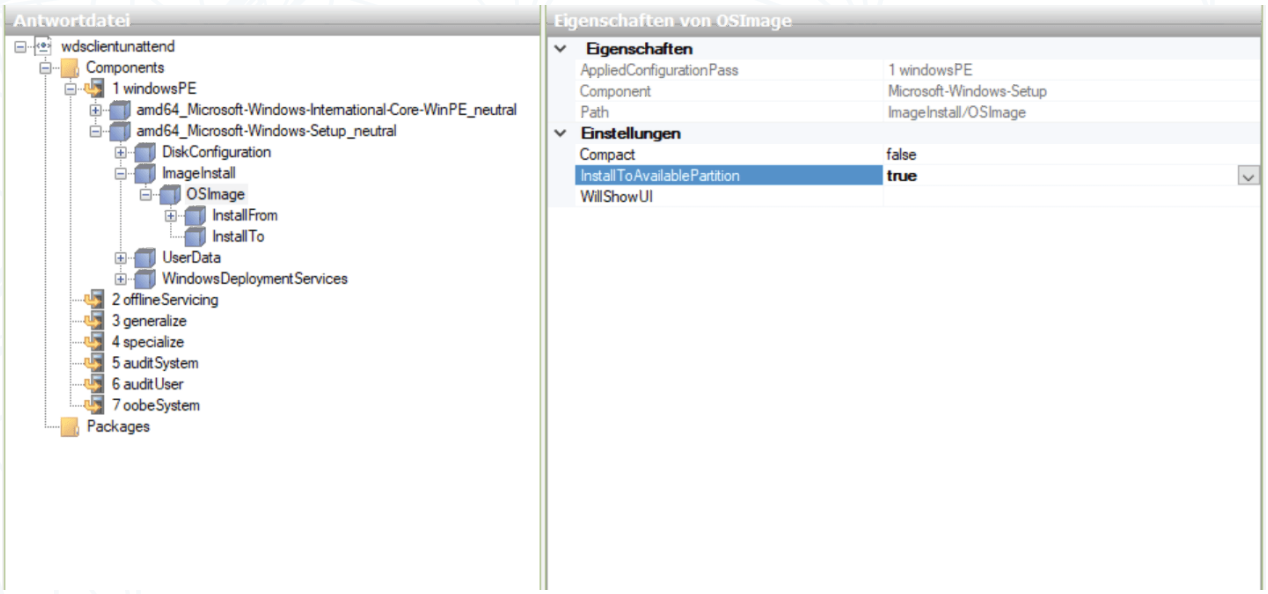
```
30          <ModifyPartitions>
31              <ModifyPartition wcm:action="add">
32                  <Order>1</Order>
33                  <Format>FAT32</Format>
34                  <Label>System</Label>
35                  <Letter>S</Letter>
36                  <PartitionID>1</PartitionID>
37              </ModifyPartition>
38              <ModifyPartition wcm:action="add">
39                  <Order>2</Order>
40                  <PartitionID>2</PartitionID>
41                  <Letter>R</Letter>
42              </ModifyPartition>
43              <ModifyPartition wcm:action="add">
44                  <Order>4</Order>
45                  <PartitionID>4</PartitionID>
46                  <Letter>W</Letter>
47                  <Label>WinRE</Label>
48                  <Format>NTFS</Format>
49              </ModifyPartition>
50              <ModifyPartition wcm:action="add">
51                  <Order>3</Order>
52                  <PartitionID>3</PartitionID>
53                  <Format>NTFS</Format>
54                  <Letter>C</Letter>
55                  <Label>Windows</Label>
56              </ModifyPartition>
57          </ModifyPartitions>
58          <DiskID>0</DiskID>
```

Werte bei der Anpassung der Partitionen

Partitionierung mittels Scripts

Wenn man das Laufwerk lieber per Script partitioniert, bevor man das Setup startet, dann kann man das Verfahren in der Antwortdatei abkürzen. In diesem Fall verzichtet man auf das Löschen der Disk, indem die oben erwähnte Einstellung *WillWipeDisk* unkonfiguriert bleibt oder den Wert *false* erhält.

Außerdem muss man unter *ImageInstall => OSImage* die Option *InstallToAvailablePartion* auf *true* setzen.

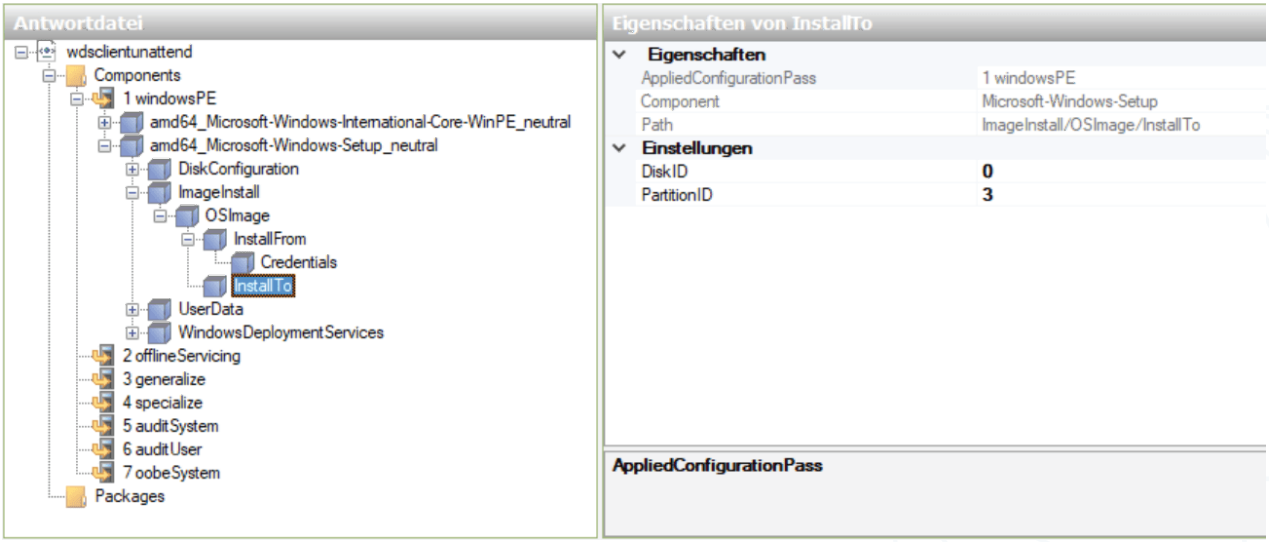


Installation in die durch ein Script eingerichteten Partitionen veranlassen

Umgekehrt muss dieser Wert auf *false* stehen, wenn man die Partitionen über die Antwortdatei anlegt.

Quelle und Ziel für die Installation festlegen

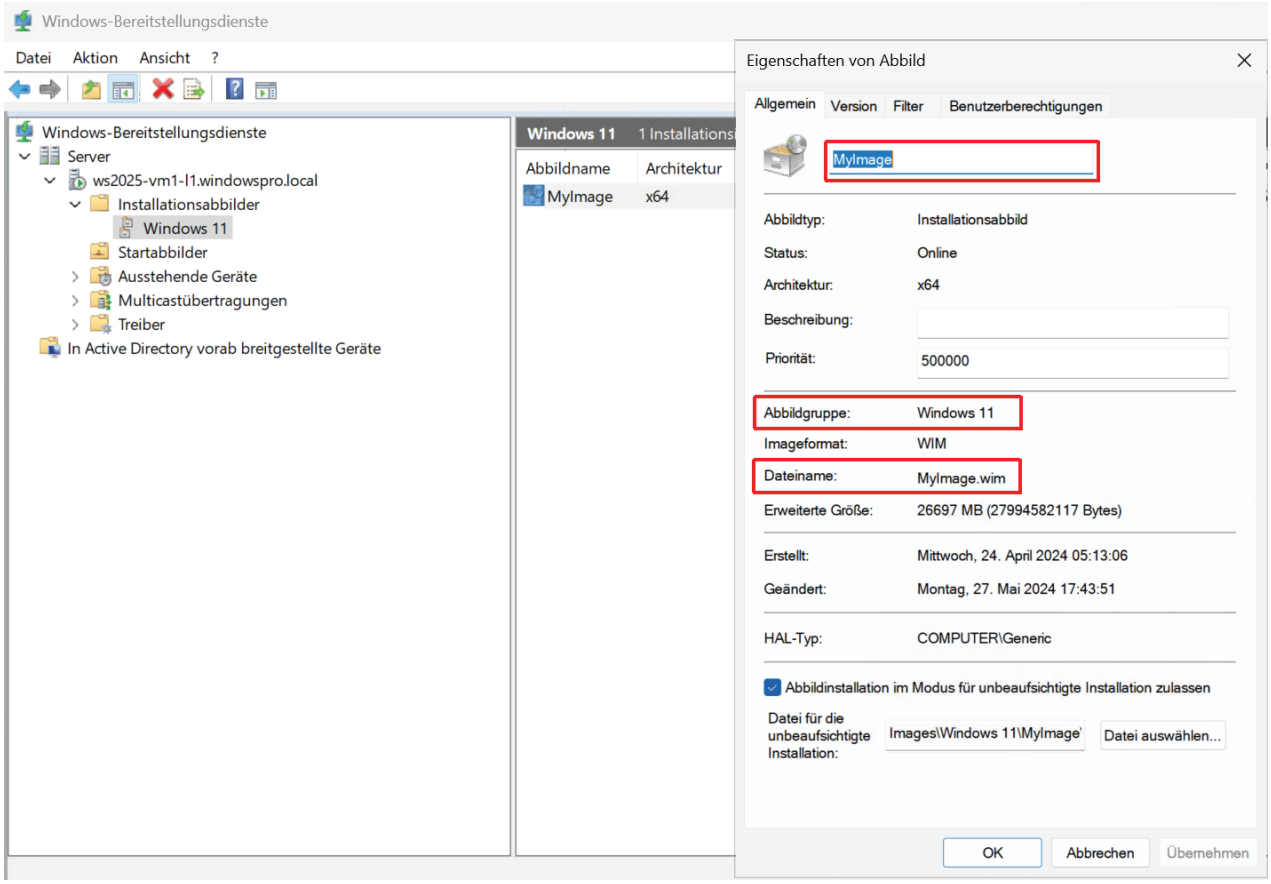
Unter *ImageInstall => OSImage => InstallFrom* teilt man dem Windows-Setup mit, welche WIM-Datei es für die Installation verwenden soll. Dabei kann es sich um ein Custom Image handeln. Unter *InstallTo* gibt man die ID der Disk sowie der Windows-Partition an, in unserem Beispiel ist das die 3.



ID der Disk und Partition angeben, in die Windows installiert werden soll

Um das Setup automatisch ablaufen zu lassen, setzt man unter *UserData* die Einstellung *AcceptEula* auf *true* und gibt unter *UserData* => *ProductKey* einen Lizenzschlüssel ein. Hier kann man einen generischen Key verwenden, wenn man die Schlüssel nachher über KMS oder ADBA zuweist.

Der Abschnitt *WindowsDeploymentServices* enthält eine eigene Einstellung für *InstallTo* sowie eine weitere zur Auswahl des Images auf dem WDS-Server. Die dafür erforderlichen Werte für *FileName*, *ImageGroup* und *ImageName* kann man den Eigenschaften eines Abbilds auf dem WDS-Server entnehmen.



Werte zur Auswahl des Images in der Antwortdatei vom WDS-Server auslesen

Dieser Abschnitt wird nur ausgeführt, wenn man eine boot.wim für Windows 10 nutzt, um Windows 11 noch weiterhin via WDS zu installieren. Starten die Rechner über ein reguläres WinPE, dann spielen diese Einstellungen keine Rolle.

Ergänzende Antwortdateien

Wenn man die Antwortdatei als *autounattend.xml* gespeichert und in das WinPE-Image übertragen hat, dann ist diese nur für die Einstellungen der beiden beschriebenen sowie einiger anderer Komponenten zuständig (die etwa der Netzwerk¬konfiguration während des Setup dienen).

Eine weitere Automatisierung der darauffolgenden Installationsphasen erreicht man durch Antwortdateien, die man in das Windows-Image integriert.

Kombiniert man die hier beschriebene *autounattend.xml* mit der *unattend.xml* für die OOBE-Phase, dann erreicht man damit bereits eine vollständig unbeaufsichtigte Installation. Weitere Einstellungen für die anderen Setup-Abschnitte kann man dann bei Bedarf in Windows SIM konfigurieren.

Fehlersuche

Wenn das Setup mit einer mehr oder weniger aussagekräftigen Fehlermeldung abbricht, dann liegt dies häufig an einer falsch konfigurierten Antwortdatei. Windows SIM prüft diese zwar auf syntaktische Korrektheit, aber lässt logische Fehler häufig durchgehen.

So kann es sein, dass man *WillWipeDisk* auf *false* gesetzt hat, aber Partitionen anlegen möchte. Oder man konfiguriert *InstallToAvailablePartion* auf *true* und kassiert einen Fehler, wenn man eigene Partitionen erstellen möchte.

In solchen Situationen kann man die Log-Files in WinPE unter *x:\windows\panther* untersuchen. Aufschlussreich ist dann meistens der Inhalt von *setuperr.log*.

Windows mit ACMP von Aagon im Netzwerk verteilen

Das OS Deployment mit den Bordmitteln erfordert, wie aus der obigen Beschreibung ersichtlich, sehr viel Aufwand, um eine Infrastruktur für eine weitgehend automatisierte Installation von Windows einzurichten. Kostenlose Frameworks wie OSDBuilder und OSDCloud beschleunigen diesen Prozess.

Allerdings handelt es sich bei diesen Tools weitgehend um Programme für die Kommandozeile und es mangelt ihnen an wesentlichen Fähigkeiten, die im Anschluss an das Rollout des Betriebssystems gefragt sind. Dazu gehört ein Reporting über den Erfolg dieses Vorgangs oder das Verteilen der erforderlichen Anwendungen.

Ausgewachsene Client-Management-Lösungen wie ACMP von Aagon bieten dagegen nicht nur eine intuitive grafische Bedienerführung inklusive Wizards für viele Aufgaben, sondern auch eine Inventarisierung, ein Reporting sowie ein Modul für die Software-Verteilung. Auch die Sicherheitsfunktionen wie Schwachstellenmanagement, Defender Management, BitLocker Management und Security Detective sind hier zu nennen.

Setup versus Golden Image

Sieht man von den Feature-Updates ab, die über Windows Update oder WSUS verteilt und dann in-place installiert werden, gibt es traditionell zwei Verfahren für die automatisierte Installation von Windows.

Die erste besteht in der Ausführung des Setup-Programms, das man mit einer Antwortdatei steuert, so dass es ohne Eingriff eines Benutzers durchläuft. Die alternative Methode besteht darin, eine Musterinstallation als WIM-Datei zu erfassen und mit DISM oder PowerShell auf die Ziel-PCs anzuwenden. Aber auch dafür benötigt man zur vollständigen Automatisierung eine Antwortdatei.

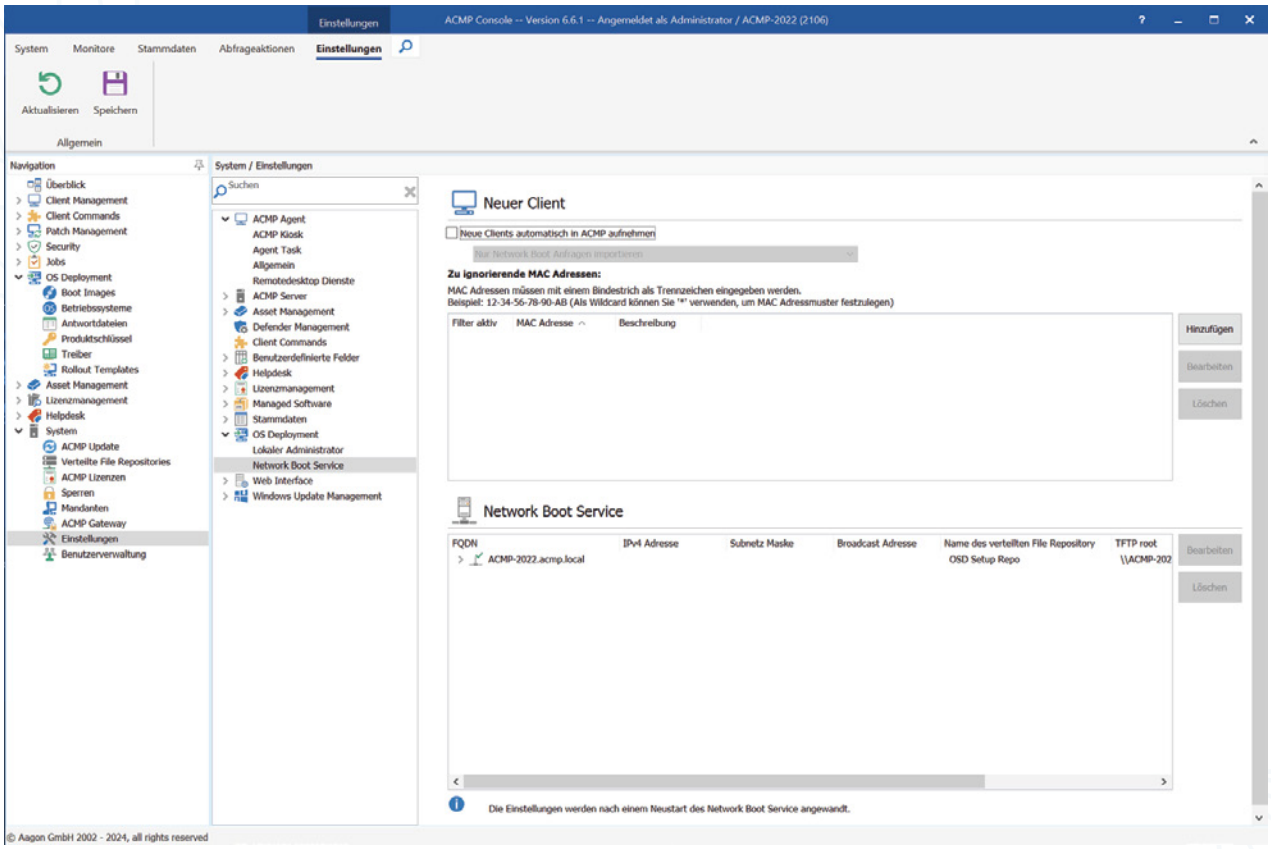
Zusätzlich benötigen beide Methoden ein Boot-Image zum Starten der PCs mittels WinPE über das Netzwerk, ein (angepasstes) Abbild des Betriebssystems, Treiber für unterschiedliche Hardware und Produktschlüssel.

ACMP verfolgt dabei einen modularen Ansatz, bei dem alle Bestandteile des Deployment-Prozesses getrennt erstellt oder bearbeitet werden. Anschließend packen Admins die für die jeweiligen Rechner vorgesehenen Komponenten in ein Template und weisen dieses den betreffenden Clients zu.

PXE und Boot-Images

Um die Installation eines Rechners zu starten, bootet man diesen zumeist über das Netzwerk, wobei dafür Windows PE zum Einsatz kommt.

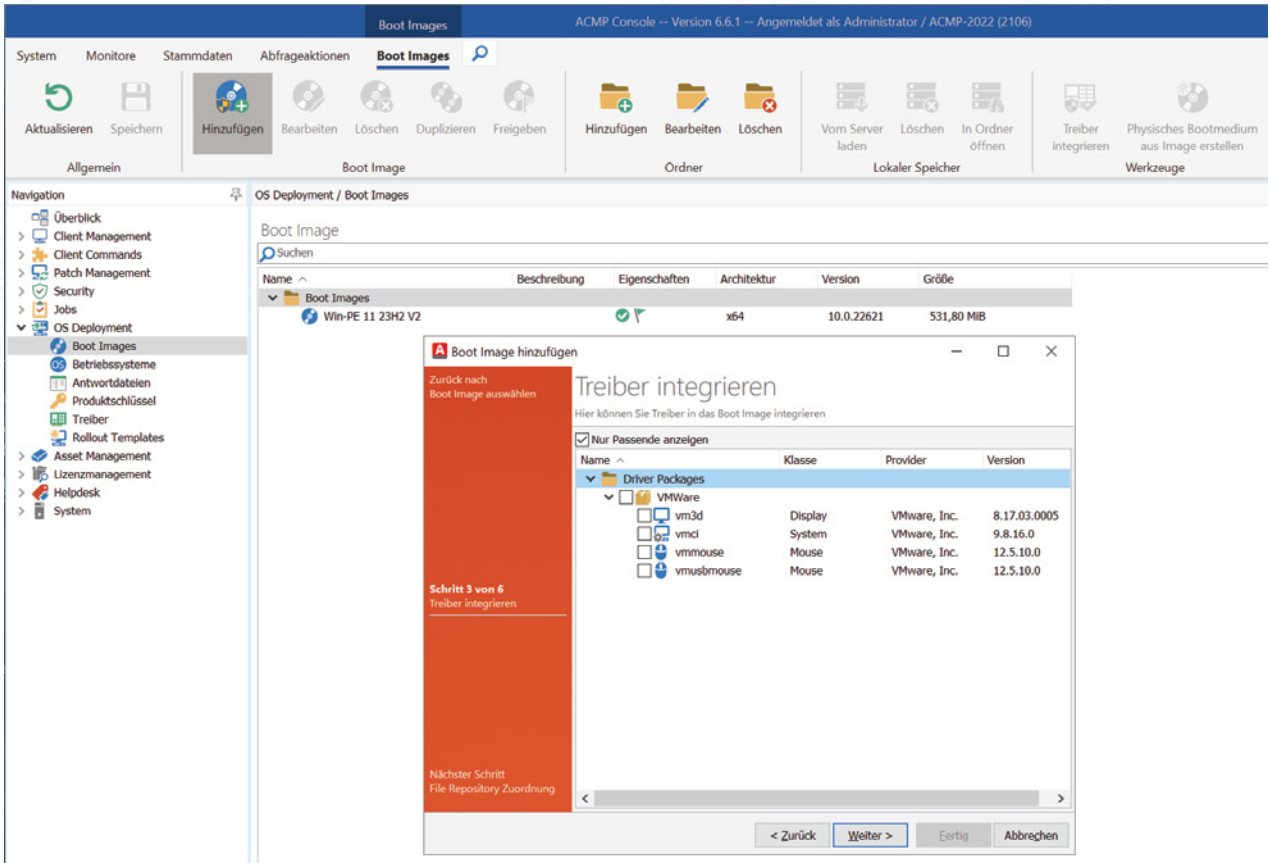
ACMP enthält zu diesem Zweck einen eigenen PXE-Server. Dieser Network Boot Service lässt sich über die Konsole konfigurieren, wobei man ihn an ein bestimmtes Interface binden und Clients auf Basis ihrer MAC-Adresse ausschließen kann.



Konfiguration des Network Boot Service in ACMP

Für die spätere Zuweisung eines OSD-Templates ist es hilfreich, wenn man neue Clients automatisch in das Management-System aufnimmt. Dies kann man abhängig von allen DHCP- oder von reinen Boot-Anfragen tun.

Das erforderliche Betriebssystem können Admins in Form mehrerer Boot-Images bereitstellen. Diese kann man zum Beispiel mit verschiedenen Treibern oder Scripts zur Vorbereitung der Installation ausstatten. Das Modul ACMP OS Deployment akzeptiert den Upload von WIM- oder ISO-Dateien, wobei es bei Letzteren das Image selbständig extrahiert.



Einem neuen Boot-Image lassen sich beim Import gleich Treiber zuweisen

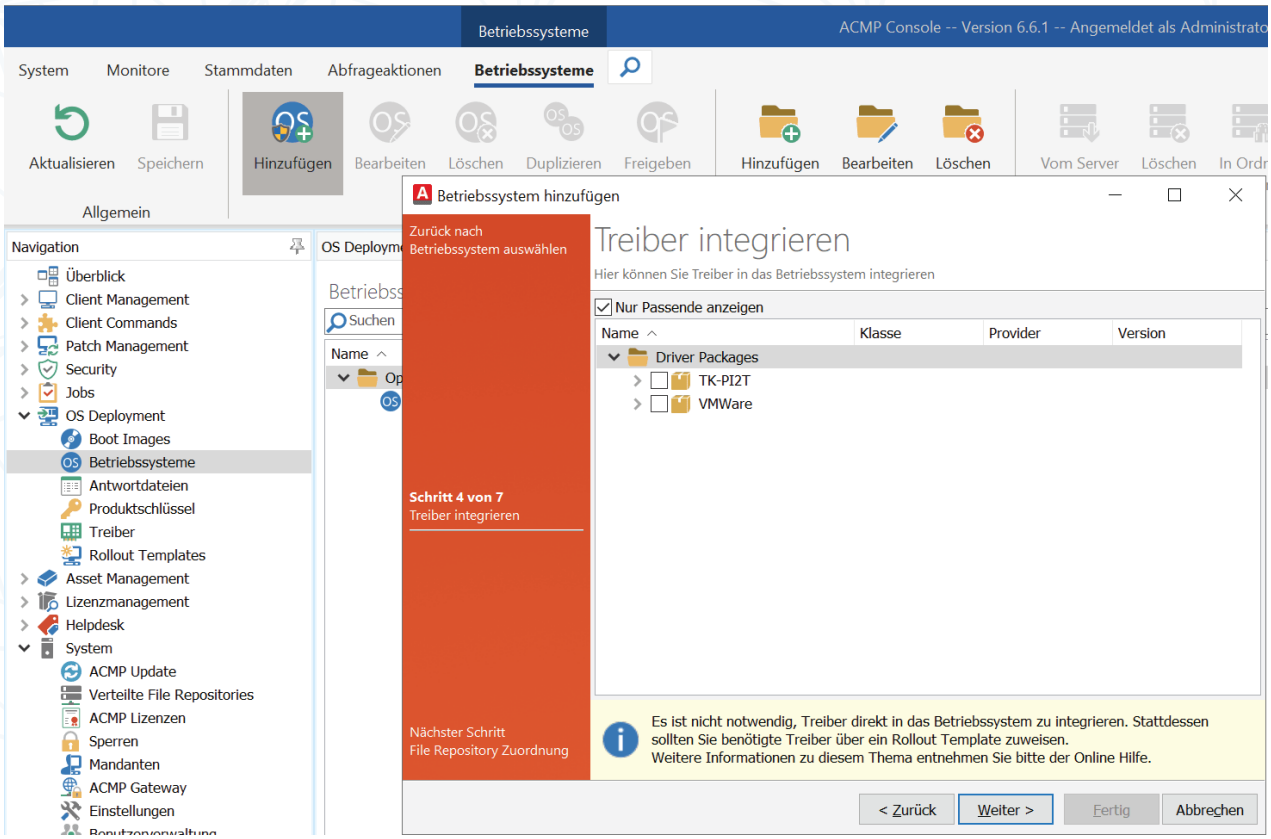
Wenn man PCs für die Installation nicht über das Netz booten möchte, dann kann man hier alternativ ein physisches Boot-Medium erstellen. Dafür muss ACMP allerdings das Windows ADK auf dem Rechner vorfinden, auf dem die ACMP Konsole läuft.

OS-Images bereitstellen

Wie bei den Boot-Images können Admins auch mehrere Abbilder für die Windows-Installation konfigurieren. Durch den Importvorgang führt ein Wizard, der im Gegensatz zu den Boot-Images nicht bloß die Integration von Treibern vorsieht.

Im ersten Dialog muss man sich entscheiden, ob die Installation mit Hilfe des Setup-Programms oder durch Anwenden eines Golden Image erfolgen soll. Diese Entscheidung wirkt sich auf die spätere Verfügbarkeit von bestimmten Optionen aus.

In der Folge wählt man die gewünschte Edition und hat die Möglichkeit, Treiber in das Image zu übernehmen. Da dies aber dem modularen Ansatz von ACMP widerspricht, rät der Dialog davon ab. Vielmehr führt man die beiden Komponenten später in einem Template zusammen.



Die Treiber für ein Image sollte man später über ein Template zuordnen

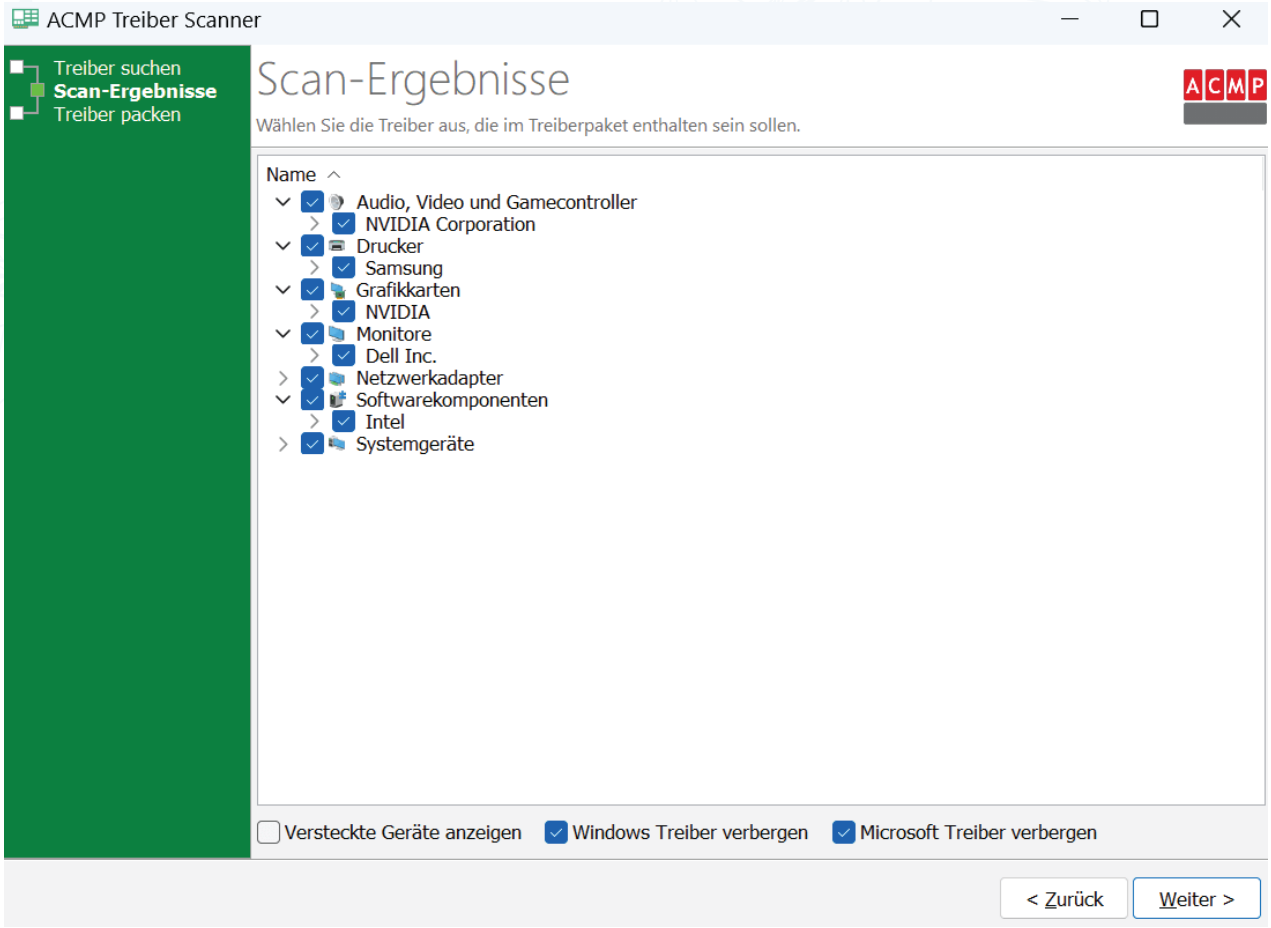
Zum Abschluss legt man fest, auf welche File-Repositories man das Image synchronisieren und ob man das Original nach dem Upload löschen möchte.

Treiber auf Muster-PC erfassen

Wenn die Hardware einen Treiber von OEMs benötigt, dann kann man diesen in den ACMP Server hochladen, indem man typischerweise die dazugehörige *.inf-Datei angibt.

Sollen aber zum Beispiel mehrere baugleiche (neue) Notebooks installiert werden, dann bietet Aagon für diese Situation einen Treiber-Scanner an. Er sucht nach allen Treibern auf einem Gerät und übernimmt die gewünschten in ein Paket, das man anschließend nach ACMP importiert.

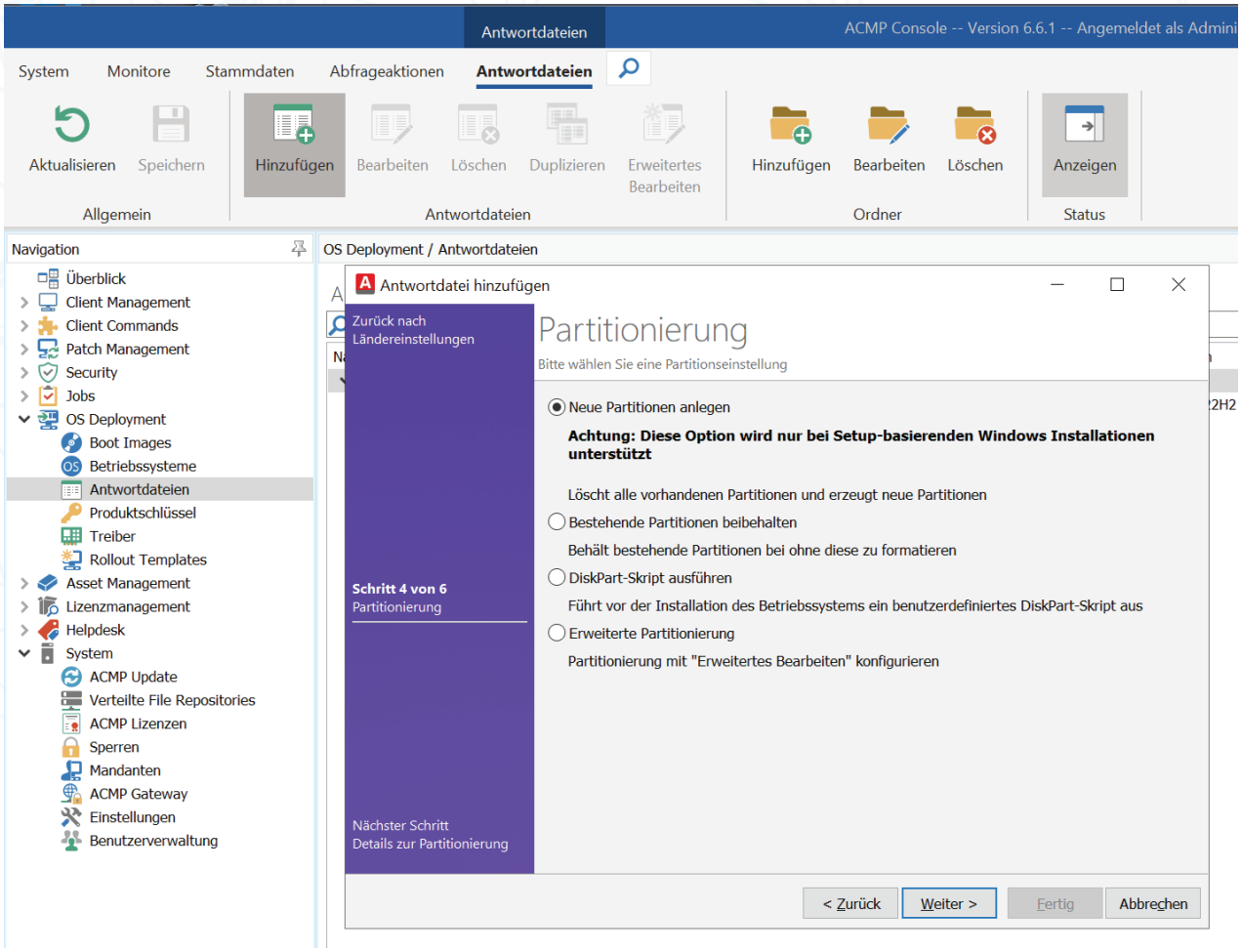
Später kann man dann beim Erstellen des Templates für ein bestimmtes Notebook-Modell alle Treiber in einem Durchgang als Paket zuordnen.



Der Scanner findet alle Treiber auf einem PC, die man dann selektiv in ein Paket übernehmen kann

Antwortdateien erstellen

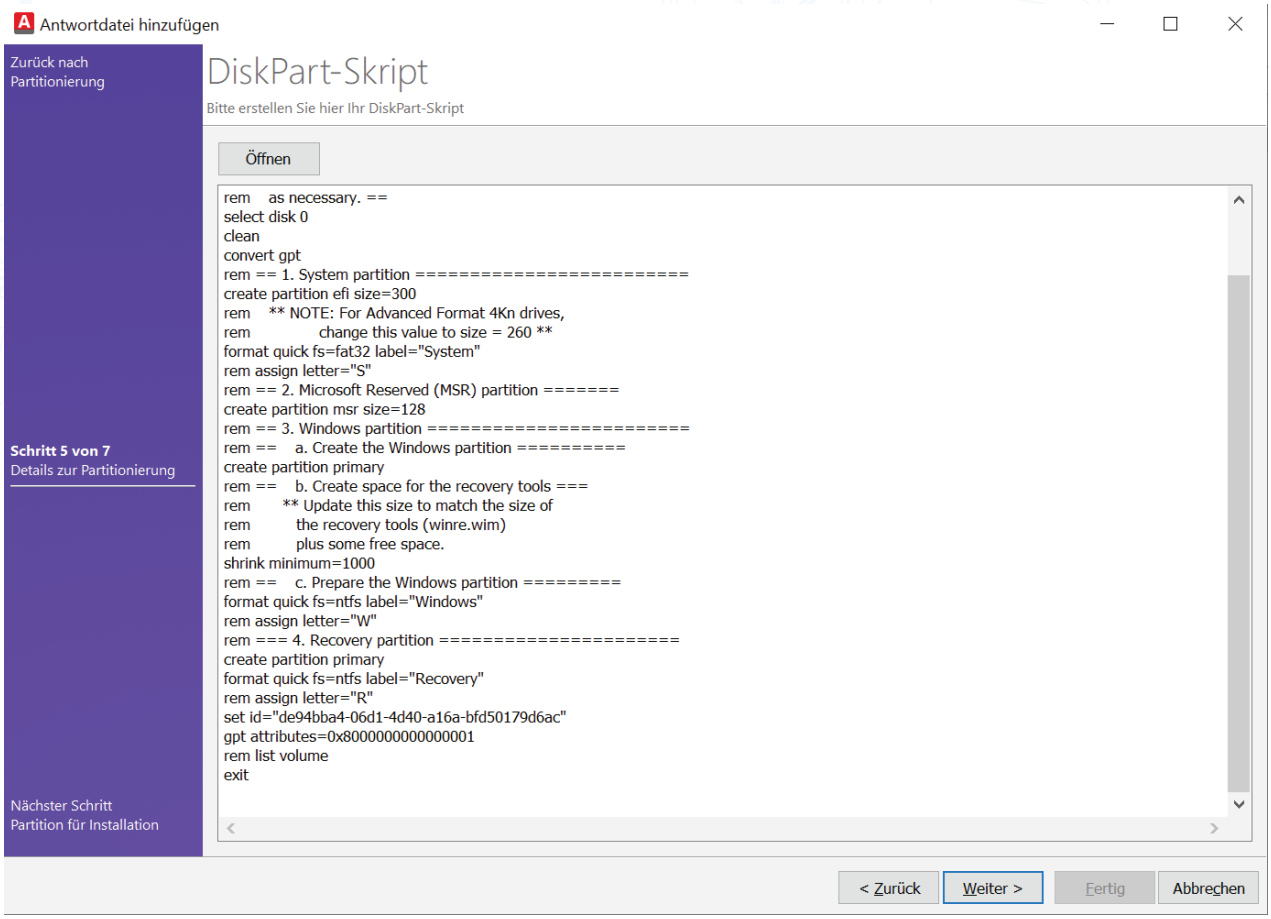
ACMP OS Deployment bietet einen Wizard, der das Anlegen einer Antwortdatei vereinfacht. Er hilft allerdings nur bei der Automatisierung der windowsPE-Phase, so dass sich damit im Wesentlichen bloß die diesbezüglichen Einstellungen (Sprache und Region, Partitionierung) konfigurieren lassen. Hinzu kommt noch die Option zum Domain-Join.



Antwortdatei mit Hilfe eines Wizards erstellen

Möchte man zusätzliche Einstellungen für die Antwortdatei konfigurieren, dann kann man anschließend über den Befehl Erweitertes Bearbeiten den Windows System Image Manager (SIM) starten (vorausgesetzt das ADK ist auf dem ACMP Rechner installiert).

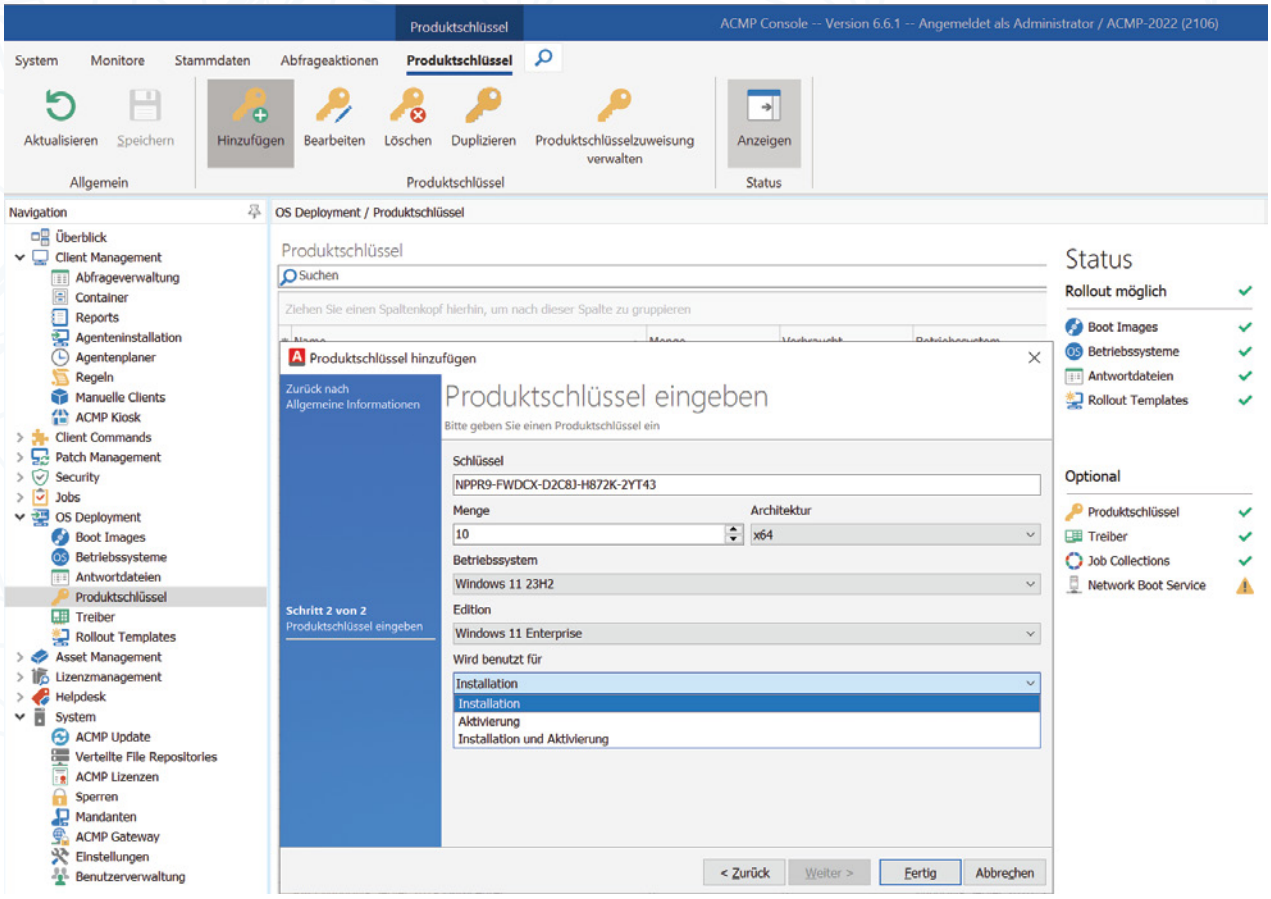
Der Dialog für die Partitionierung der Disk weist darauf hin, dass die Antwortdatei diese Aufgabe nur für das Setup-basierte Verfahren übernehmen kann. Für das Anwenden eines Golden Images muss man ein Diskpart-Script ausführen, das in ACMP bereits enthalten ist.



ACMP enthält das von Microsoft stammende Diskpart-Script für die Laufwerkspartitionierung

Produktschlüssel

Die automatische Installation von Windows lässt sich schließlich durch eine weitere Information ergänzen, nämlich dem Product Key.



Produktschlüssel für Windows 11 Enterprise in ACMP hinterlegen

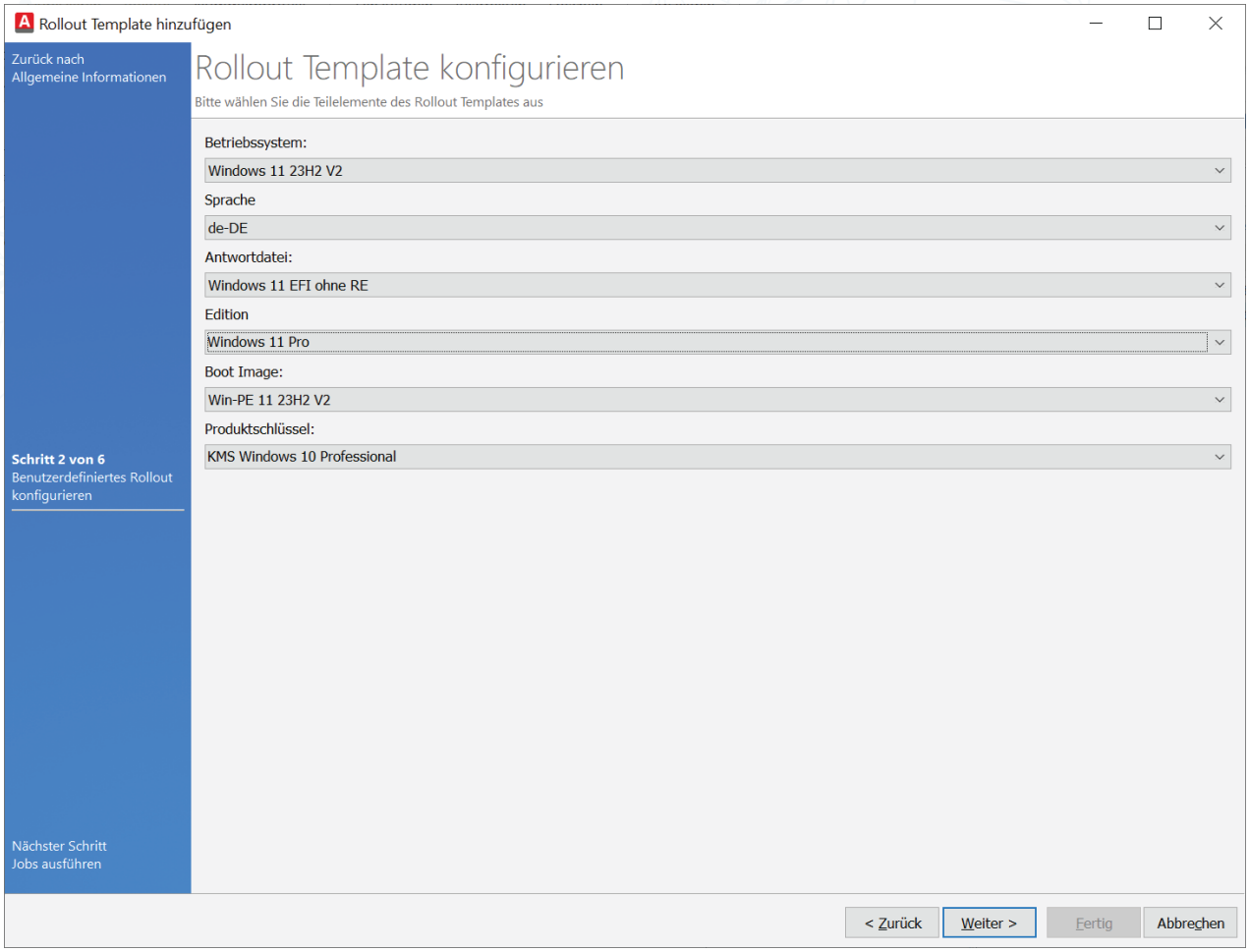
Die Produktschlüssel kann man, wie die anderen Komponenten eines Setup-Pakets, in ACMP hinterlegen. Wenn man diese später ohnehin über KMS oder ADBA zuweist, reichen hier auch die von Microsoft zur Verfügung gestellten generischen Schlüssel.

Rollout-Template erstellen

Ein Rollout-Template ist quasi ein Installationspaket, das die verschiedenen Komponenten für das OS Deployment bündelt. Aufgrund dieses modularen Ansatzes ließe sich das gleiche Image oder Treiberpaket in verschiedenen Kombinationen verwenden.

So könnte man etwa für ein bestimmtes Notebook-Modell separate Templates für die Marketing- und die IT-Abteilung zusammenstellen. Das eine enthielte dann zum Beispiel Windows 11 Pro und das andere die Enterprise Edition. Beide würden aber die gleichen Treiber nutzen.

Das Anlegen eines Templates erfolgt ebenfalls über einen Wizard. Dieser erlaubt eingangs die Installation des ACMP Agents und eines Zertifikats für das ACMP Gateway. Das eigentliche Paket wird dann im nächsten Schritt mit der Auswahl der Komponenten geschnürt.



Auswahl der Images, Antwortdatei, Edition und Produktschlüssel für ein Rollout-Template

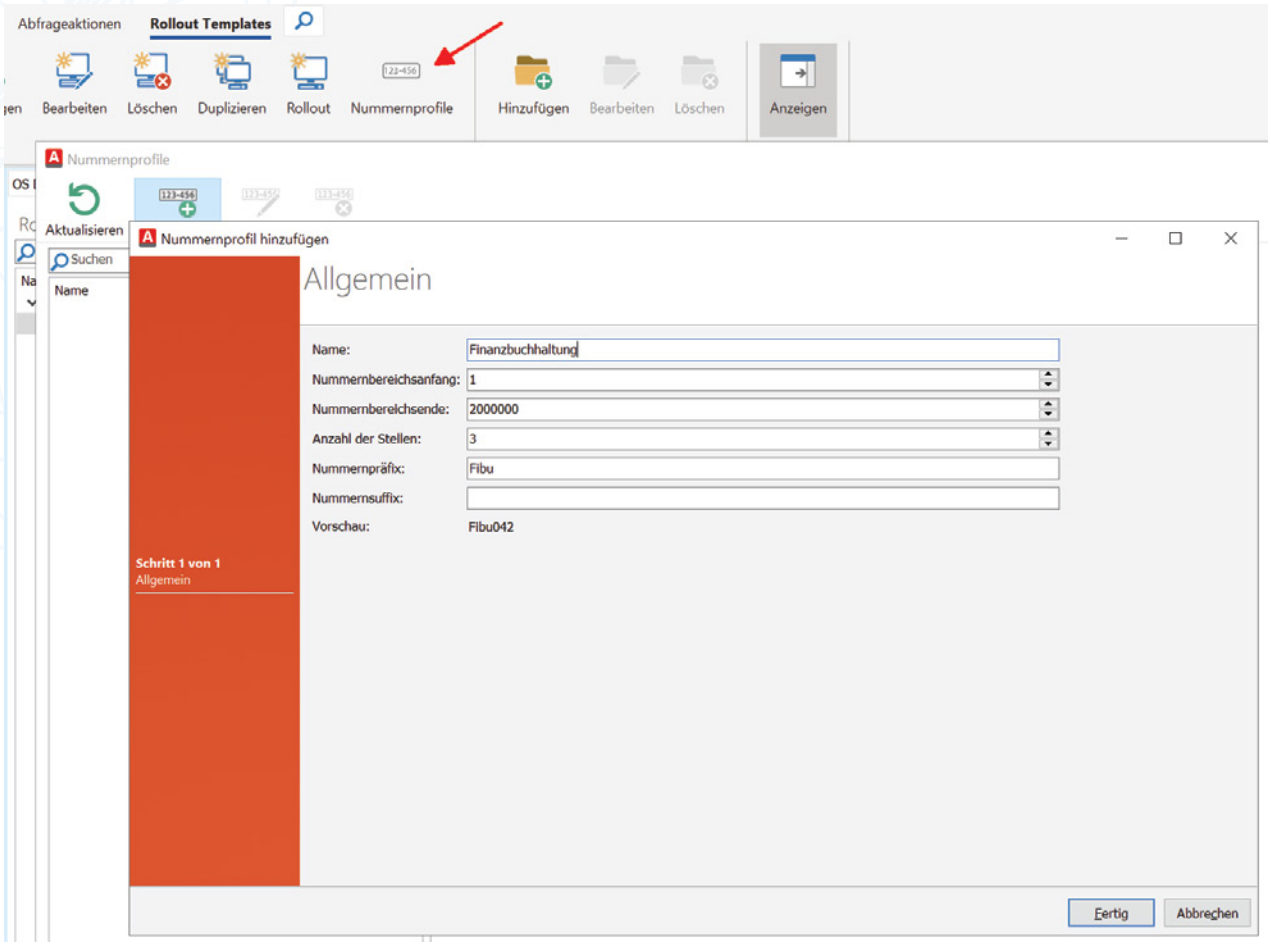
Zusätzliche Treiber kann man an dieser Stelle noch nicht hinzufügen, dafür ist ein späterer Dialog zuständig. Davor erhält man aber noch die Möglichkeit, mehr oder weniger beliebige Jobs an das Deployment anzuhängen. In erster Linie wird man damit die automatische Installation von Software, automatische Konfigurationen und Windows Updates nach Abschluss des Setups veranlassen. Hierbei wird auf die Funktionen der Module ACMP Desktop Automation (Softwareverteilung), Managed Software (3rd Party Patch Management) und ACMP CAWUM (vollständiger WSUS-Ersatz) zugegriffen.

Templates an Clients zuweisen

Um den Rollout des Betriebssystems zu starten, ordnet man das Template den gewünschten Clients zu. Dies kann man tun, indem man einzelne Geräte auswählt oder eine (dynamische) Gruppe dafür heranzieht.

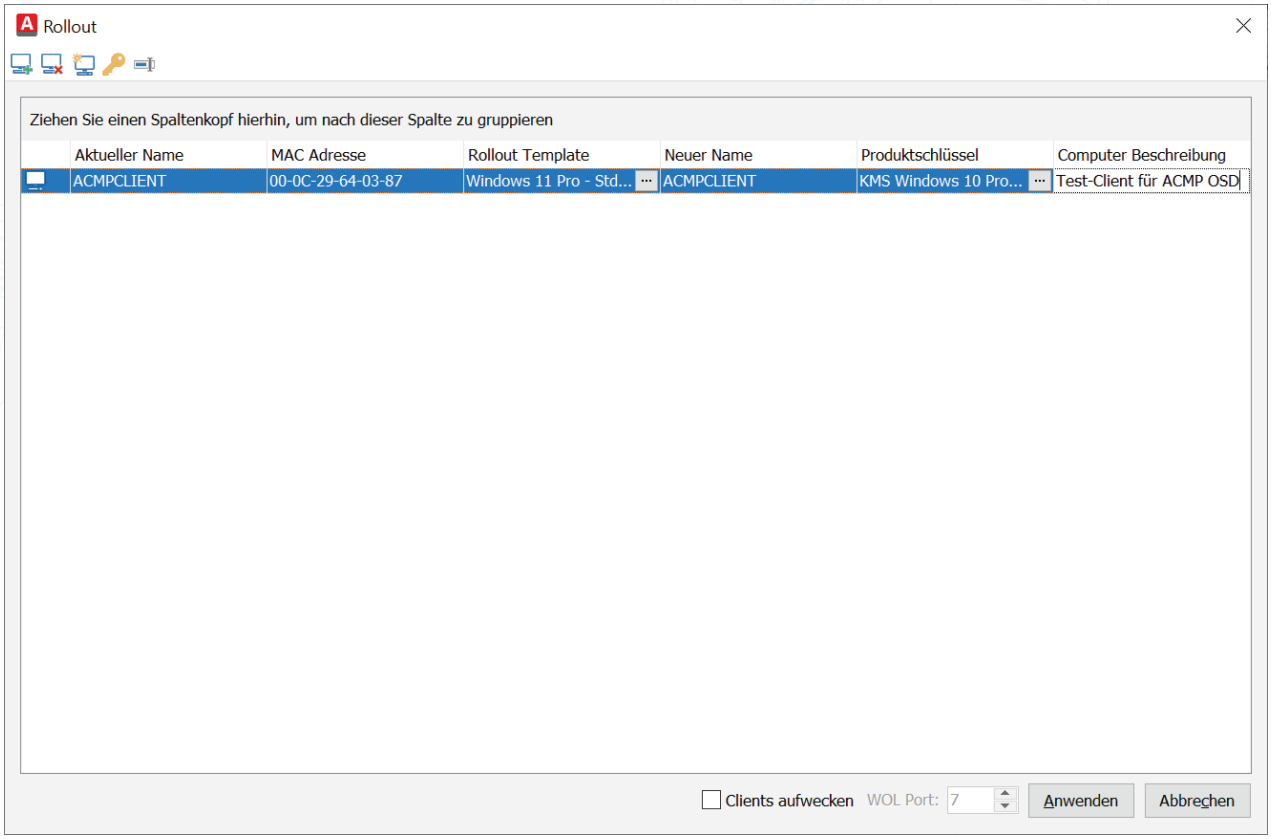
Der für die Zuweisung vorgesehene Dialog füllt nach der Auswahl des Templates selbständig die Felder für den Produktschlüssel und die MAC-Adresse aus.

Hier besteht zudem die Möglichkeit, einen automatisch generierten Namen für den Rechner zu vergeben, der einem selbst gewählten Schema folgt. Nummernprofile sind vor allem dann praktisch, wenn man eine größere Zahl an PCs installiert, weil ACMP dann die Namen einfach hochzählt.



Konfiguration eines Schemas zur Vergabe von Rechnernamen

Bei Bedarf kann man an dieser Stelle noch eine Beschreibung für den Client hinterlegen. Zusätzlich bietet dieser Dialog eine Option für das Hochfahren der Clients über Wake-on-LAN.



Rollout-Template an einen Client zuweisen

Das Abschicken dieses Formulars aktiviert die entsprechenden Jobs für die Installation von Windows auf den gewählten Geräten.

Fazit

Die Client-Management-Lösung ACMP bietet ein Modul für die automatisierte Installation von Windows, die Admins bei dieser komplexen und wiederkehrenden Aufgabe entlastet. Der Charme des Tools besteht darin, dass es alle Bausteine für das OS Deployment als eigenständige Komponenten verwaltet, die Admins dann mit Hilfe von Templates beliebig kombinieren und wieder¬verwenden können.

Dieser Ansatz vereinfacht somit das Management von Images, Produktschlüsseln und Antwort-dateien. Letztere lassen sich mit Hilfe eines Wizards generieren, wobei man allerdings für eine weitergehende Konfiguration auf Windows SIM ausweichen muss. Sehr hilfreich erweist sich zudem der Treiber-Scanner, der alle Treiber aus einer vorhandenen Windows-Installation extrahieren kann.

Das ACMP OS Deployment unterstützt sowohl die Automatisierung einer Setup-basierten Installation als auch das Applizieren eines individuellen Images, das mit DISM erfasst wurde. Der Hersteller empfiehlt jedoch das erste Verfahren, weil Admins beim zweiten eine Reihe von Aufgaben selbst erledigen müssen, um die sich sonst das Setup kümmert.

Verfügbarkeit

Neben dem OS Deployment bietet Aagon in der ACMP Konsole noch weitere Module für alle gängigen Aufgaben des Client-Managements an, darunter unter anderem die Inventarisierung, das Patch- und BitLocker-Management, die Software-Verteilung sowie einen Helpdesk. Eine vollständige Übersicht findet sich auf der Aagon-Website.

Dort können Interessenten eine kostenlose und vollumfängliche Testversion von ACMP anfordern.

ÜBER AAGON

„Manage any device in a connected world!" – Aagon entwickelt seit 30 Jahren Client-Management- und -Automation-Lösungen und ist der Spezialist für die Verwaltung von Endgeräten und die Automatisierung von Standardaufgaben. Durch sorgfältige Entwicklungen, mehr als 20 Jahre Marktreife und die enge Zusammenarbeit mit unseren Kunden und Partnern sind unsere Produkte perfekt auf Ihre Anforderungen und Bedürfnisse zugeschnitten.

Individuelle Beratung und die beste Unterstützung von Kunden und Partnern bei der Installation und ersten Einrichtung gehören deshalb zum Standard von Aagon. Ein umfassendes Verständnis von Kundenbedürfnissen und der ständige Kontakt zu unseren Kunden und Partnern ermöglichen Softwareentwicklung auf Augenhöhe.

Webinare-on-Demand, zahlreiche Whitepaper und die beliebten Treffen zum Anwendertreffen an Standorten in ganz Deutschland sind nur drei Beispiele, wie nahe am Kunden ACMP wirklich entwickelt wird.

Aus Gründen der besseren Lesbarkeit wird auf die gleichzeitige Verwendung der Sprachformen männlich, weiblich und divers (m/w/d) verzichtet. Sämtliche Personenbezeichnungen gelten gleichermaßen für alle Geschlechter.

EIN PRODUKT DER
Aagon GmbH
Lange Wende 33
D-59494 Soest
Fon: +49 (0)2921 - 789200
Fax: +49 (0)2921 - 789244
sales@aagon.com
www.aagon.com

GOLD
UNIFIED ENDPOINT MANAGEMENT
IP INSIDER
AWARD 2023

techconsult
TECHNOLOGY MARKET ANALYSTS
ein Unternehmen der heise group

PUR 2024
Professional User Rating
★ 5-FACH ★
★ CHAMPION ★