

FireGeier Unattended Vista Guide

<http://www.msfn.org/board/index.php?showtopic=95462>

<http://firegeier.unattended-sponsor.de/en/introduction.html>

pdf build by Emile

BASICS

Introduction

- Tools
- Important note for WDS installations
- If something is going wrong
- Recommendation for further experimentation

Install Tools

- 1. MS WAIK (Windows Automated Installation Kit)
- 2. SETX.exe
- 3. VU-Batch-Module
 - Installation of VU Batch Modules
 - Executing PrepLab.cmd
- 4. Tools we can install optional

VU Batch Modules

- How to use VU Batch Modules
- The batch files
 - PrepLab.cmd
 - image_info_boot_wim.cmd and image_info_install_wim.cmd
 - mount_boot_wim.cmd and mount_install_wim.cmd
 - unmount.cmd
 - unmount_commit.cmd
 - create_iso.cmd
 - au_fehler.cmd
 - build_menu.cmd
 - check_umgebung.cmd

BEGINNERS

HowTo: Creating an answer file (Autounattend.xml) using WSIM

- Step 1: Opening WSIM
- Step 2: Opening Windows image file (install.wim)
- Step 3: Creating new answer file
- Step 4: Add components from catalog to answer file
- Step 5: Editing component settings in answer file
 - windowsPE \ Microsoft-Windows-International-Core-WinPE
 - windowsPE \ Microsoft-Windows-International-Core-WinPE \ SetupUILanguage
 - windowsPE \ Microsoft-Windows-Setup \ Display
 - Refresh Rate problem
 - How to set display for end users desktop?
 - windowsPE \ Windows-Setup \ ImageInstall \ OSImage
 - windowsPE \ Windows-Setup \ ImageInstall \ OSImage \ InstallFrom
 - windowsPE \ Windows-Setup \ ImageInstall \ OSImage \ InstallFrom \ MetaData
 - What does that mean?
 - Where we can get the Value from?
 - Option 1: Use /IMAGE/NAME
 - Option 2: Use /IMAGE/INDEX
 - Option 2: Use /IMAGE/INDEX
 - windowsPE \ Windows-Setup \ UserData
 - windowsPE \ Windows-Setup \ UserData \ ProductKey
 - specialize \ Windows-Security-Licensing-SLC-UX
 - oobeSystem \ Windows-International-Core
 - oobeSystem \ Windows-Shell-Setup
 - oobeSystem \ Windows-Shell-Setup \ AutoLogon
 - oobeSystem \ Windows-Shell-Setup \ Display
 - Refresh Rate problem
 - How to set display for end users desktop?
 - oobeSystem \ Windows-Shell-Setup \ OOBE

- OOBE background information
 - What is OOBE?
 - What is SkipMachineOOBE property doing?
 - Why SkipMachineOOBE is deprecated?
 - How to bypass OOBE properly - how to automate OOBE?
 - Known issues automating OOBE
- Step 6: Validating answer file
- Step 7: Saving answer file as Autounattend.xml
- Step 8: Copying Autounattend.xml to removable media (Floppy, USB Flash Drive, DVD)
- Step 9: Executing Vista setup
- Known issue: Autounattend.xml seems not to be executed

HowTo: Partitioning

- Step 1: Add DiskConfiguration components to Autounattend.xml
- Step 2: Setting Up Disk Configuration Properties
 - What is a Partition?
 - What is the use of Partitions?
 - Partition Types
 - Primary Partitions
 - Extended Partitions and Logical Drives
 - Setting up CreatePartition and ModifyPartition
 - CreatePartition
 - windowsPE \ Windows-Shell-Setup \ DiskConfiguration \ Disk("O")
 - CreatePartition Entry 1
 - CreatePartition Entry 2
 - CreatePartition Entry 3
 - Size or Extend
 - ModifyPartition
 - ModifyPartition Entry 1
 - ModifyPartition Entry 2
 - Avoid tripping into "Extend trap"
 - Configuring InstallTo & InstallFrom
 - OSImage
 - InstallFrom
 - MetaData
 - InstallTo
- Limitations
- Summary
- Notes for WDS installations

Partitioning XML snippets

- 1. Formatting first, already existing partition
- 2. Delete all partitions and create a new one
- 3. Delete all partitions and create two new partitions
- 4. Delete all partitions and create three new partitions

HowTo: Creating ISO file

- Step 1: Putting all files and folders into one directory on HD
- Step 2: Create ISO file
- Step 3: Burn ISO onto DVD media using third party software
- Known issue: Autounattend.xml will not to be executed

ADVANCED

HowTo: Injecting drivers into image (install.wim)

- Step 1: Create new unattend.xml using WSIM
- Step 2: Mount install.wim
- Step 3: Activating advanced log function for driver installations
- Step 4: Execute Packagemanager
- Step 5: Unmount install.wim
- Step 6: Create ISO file

HowTo: Installing drivers from media (USB Stick, DVD etc.) directly

- Step 1: Add Driver Path to Windows PE pass of your Autounattend.xml
- Step 2: Mounting boot.wim
- Step 3: Creating winpeshl.ini
- Step 4: Creating the SetDirversRoot.cmd
- Step 5: Unmount boot.wim saving changes
- Step 6: Creating drivers directory
- Step 7: Copying files and folder to removable media
 - Option 1: Copying to USB flash drive
 - Option 2: Copying to DVD
- FAQ

HowTo: Slipstreaming updates into install.wim

- Alternative I - Manual Integration
 - Step 1: Copying updates into one folder
 - Step 2: Mounting install.wim
 - Step 3: Creating temporary folder
 - Step 4: Extracting update files
 - Step 5: Importing update packages to WSIM
 - Step 6: Adding update and hotfix packages to answer file
 - Step 7: Saving answer file as Integrate.xml
 - Step 8: Integrating updates using package manager pkgmgr.exe
 - Step 9: Save changes to install.wim
 - Step 10: Clearing Sandbox directory
 - If something went wrong
- Alternative II - Integrating updates using a batch
 - offline_update.cmd

HowTo: Installing applications during unattended Vista setup

- Step 1: Creating an Install folder on removable media
- Step 2: Editing answer file (Autounattend.xml)
 - Examples
 - Regtweaks.reg
 - Adobe Reader (extracted already)
 - Mozilla Firefox
 - Adobe Flashplayer
 - Sun Jave VM
 - Leave audit modes and boot to oobe
- How to do a reboot between applications installs?

HowTo: Importing Registry Tweaks during Vista setup

- The structure of Regtweaks.reg
- Step 1: Creating and saving a Regtweaks.reg
- Step 2: Preparing the registry import of Regtweaks.reg using WSIM
- How to apply registry tweaks for all further user profiles in future (Copying to Default User)?

HowTo: Copying the actual user profile over to the profile of default user

- Step 1: Executing sysprep.exe automatically for generalization
- Step 2: Checking over the settings in Autounattend.xml
 - specialize \ Windows-Deployment_neutral \ RunSynchronoursCommand 1
 - specialize \ Windows-Deployment_neutral \ RunSynchronoursCommand 2
 - auditSystem \ Windows-Shell-Setup \ AutoLogon
- Step 3: Adding additional Settings
 - generalize \ Windows-PnPSysprep
 - specialize \ Windows-Security-Licensing-SLC-UX
 - specialize \ Windows-Shell-Setup
 - oobeSystem \ Windows-International-Core
 - oobeSystem \ Windows-Shell-Setup \ Display
 - oobeSystem \ Windows-Shell-Setup \ OOBE
 - oobeSystem \ Windows-Shell-Setup \ UserAccounts \ LocalAccounts

DOWNLOAD

Download VU Batch Modules

http://firegeier.unattended-sponsor.de/files/en/vu_batchs.exe

MD5: EC26669872967DDE7E632E0F334A0E1D

SAH1: 4D46B943531696E5B9070FF452B34CBD1BBC594D

ABOUT

Contact

<http://firegeier.unattended-sponsor.de/en/contact.html>

BASICS

Introduction

Tools

Microsoft has completely worked over the management tools for unattended setups and rolled out a new software package - the **Business Desktop Deployment Tools 2007 (short BDD 2007)**.

Part of this package is the **Windows Automated Installation Kit (short WAIK)**. The WAIK does contain the **Windows System Image Manager (short WSIM)**, the unattended setup reference documentations and a few very helpful command line tools.

For the further steps using this guide the installation of **BDD 2007 is not a must**. It's enough just to download and install the **WAIK**.

Using **WSIM** we will **build an answer file** on the following sites, which is **called Autounattend.xml** and **displaces** the known **winnt.sif** of older Windows versions.

Important note for WDS installations

If you want to **deploy Vista using WDS**, you will **need TWO xml answer files**. You need to have an **Autounattend.xml** doing the **configurations during Vista setup** and you need to have a **WDSClientUnattend.xml** you have to place in **RemoteInstall\WDSClientUnattend** folder of your **WDS server** to **"start"** from WDS.

I do not have a server environment, so I do not have any experience deploying Vista over WDS. There is an **introducing WDS guide** called **WDSOOBSTEPBYSTEP.DOC** located in **WDS directory** of **WAIK-CD**.

If something is going wrong

Vista setup does create very **detailed log files** - **setuperr.log** and **setupact.log**. **Former** does **log errors** only that have **caused an abort**, **latter** will **watch all setup activity**.

You will find different logs labeled like describe on your HD. The **ones** which are **relevant predominantly** are **located in**

%WINDIR%\Panther\UnattendGC.

NOTE:

Depending on the account you're logged on, you may have to **change the permissions** of the files - **right clicking them - to view the contents!**

Recommendation for further experimentation

1. Edit your answer file **using WSIM basically**. This will better **avoid syntax errors** inside your answer file.

2. Adapt changes of answer file **step by step only**. **This will make it easier to detect any errors!** **I bet your unattend setup will not run properly, if you set all kind of properties you want to use finally!!!**

Now have fun 'n' success exploring Vista unattended setup guide!

Regards,
Martin (FireGeier)

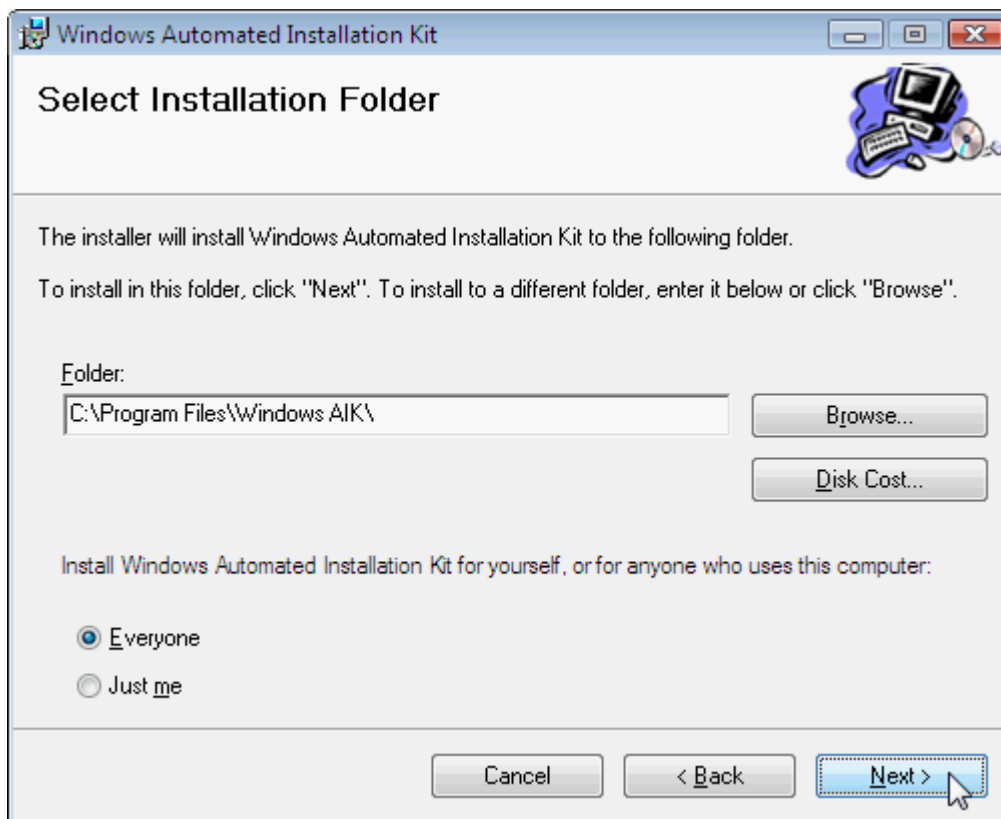
Install Tools

What we need to have installed

1. MS WAIK (Windows Automated Installation Kit)

The **WAIK** holds all tools we need to build an unattended setup. It's installed to **%Programfiles%\Windows AIK** by default. **WAIK** requires **Windows XP SP 2, Windows 2003 Server** or **Vista** operating system!

During installation of **WAIK** we keep the default settings for path:



2. SETX.exe

SETX.exe is a **command line tool** to set environment variables during a running windows session. We will create some environment variables to make access to our working directory easier.

Vista

If you use Vista to build unattended setups you **don't need to do any further steps**. SETX.exe is **included in Vista** already.

Windows XP

If you're using **Windows XP** you can **copy SETX.exe** from **MS Support Tools for XP Service Pack 2**.

You **don't need to install** the **whole** Support Tools **package**. It's enough **just** to **extract** the Support Tools exe file **to any directory** by using a tool like winrar for example. After that just **copy** the **SETX.exe** to **%Windir%\System32** directory.

3. VU-Batch-Module

The **VU Batch Modules (Vista Unattended Batch Modules)** are a collection of batch files I build, to make work of circular processes more handy.

A lot of steps we have to do to build an unattended setup are done by command line tools. So it's more convenient to do these steps just clicking a batch file instead of typing very long syntaxes again and again.

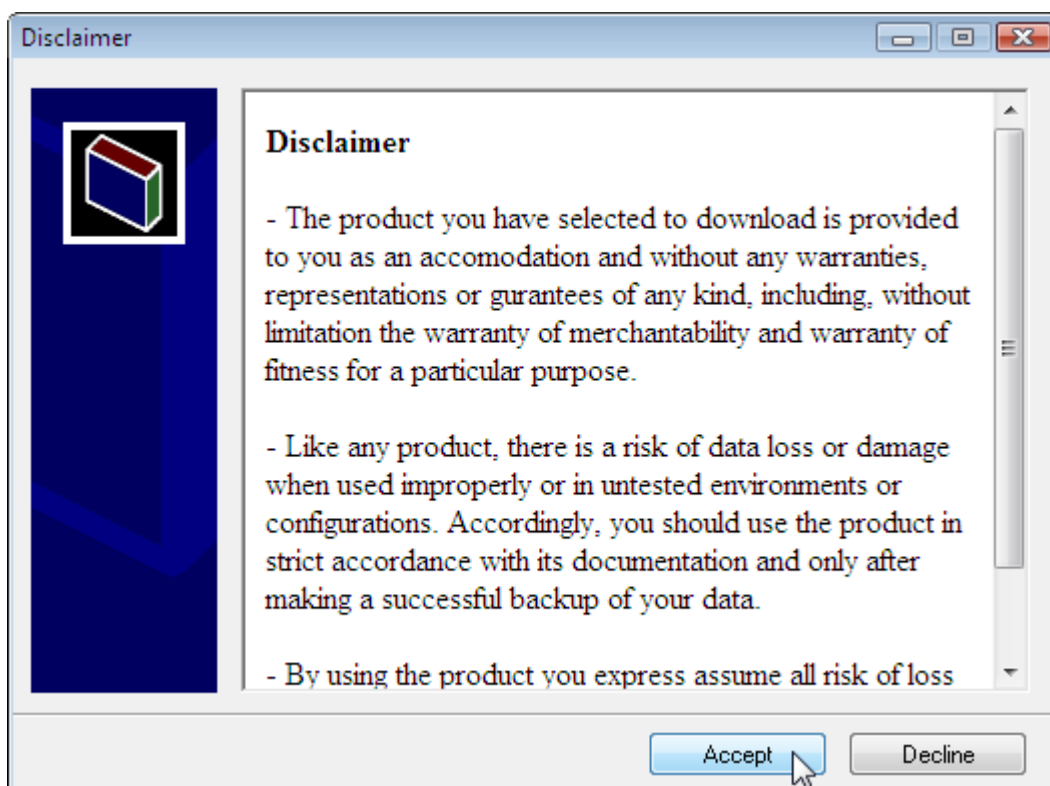
[Click here to download VU Batch Modules](#)

NOTE:

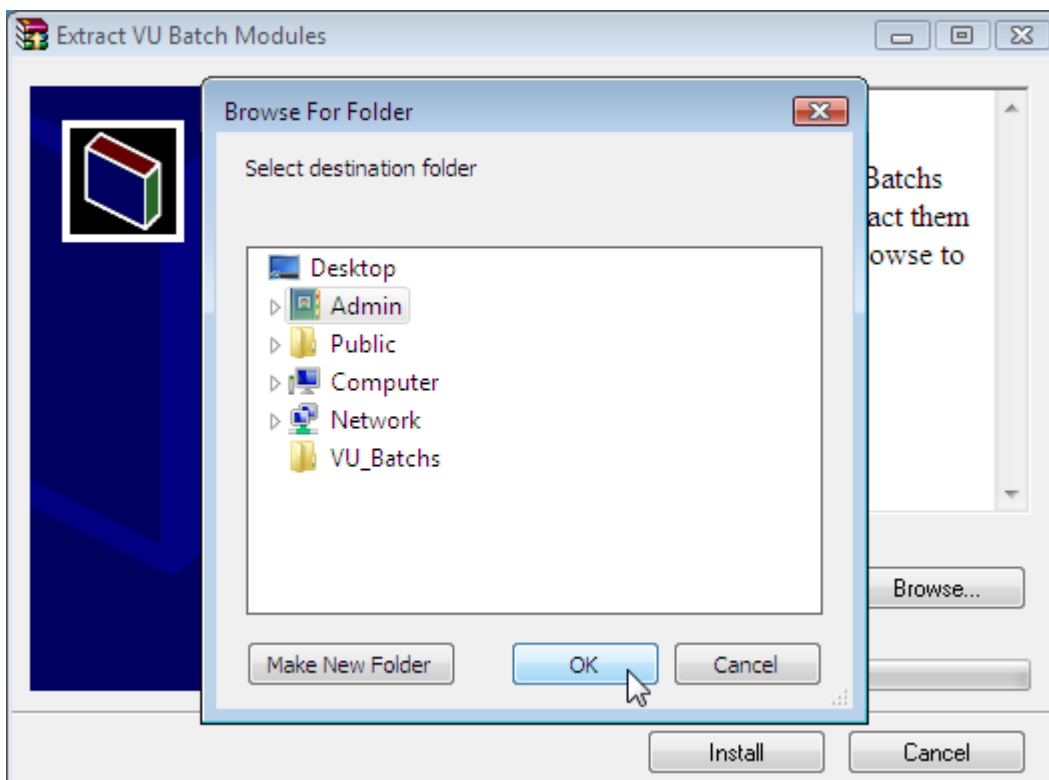
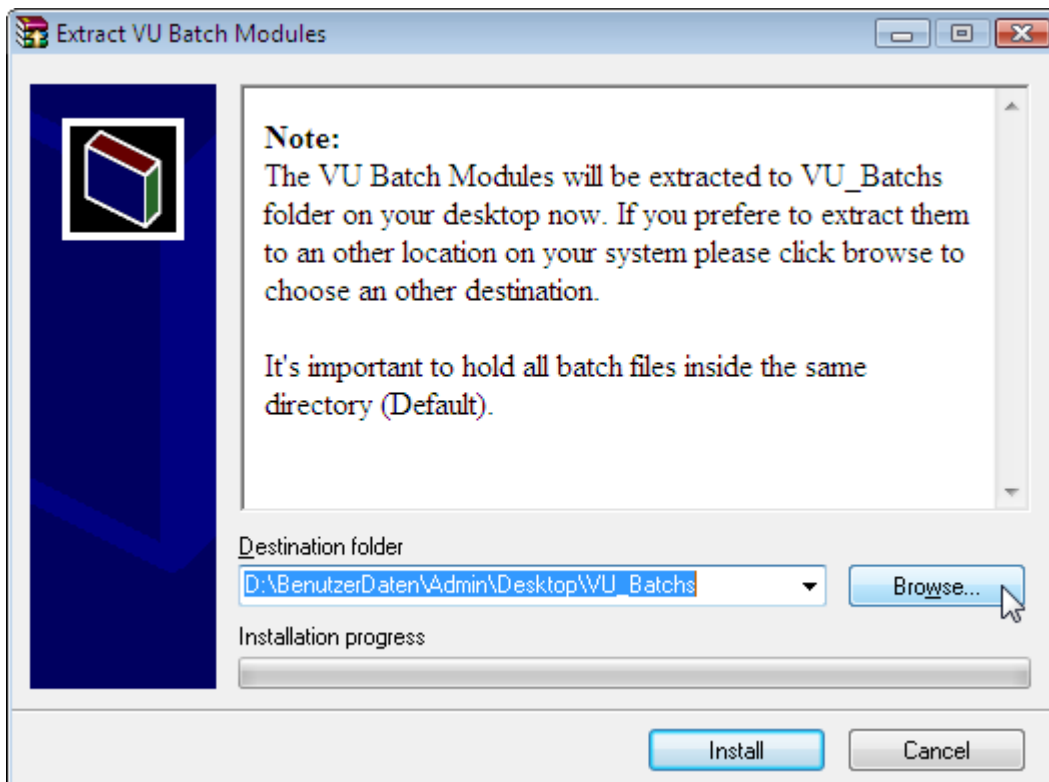
It's not a must, to use batch files to automate the command line tools. You could do the same thing with vbs for example or any other language supported by scripting host.

Installation of VU Batch Modules

After downloading **vu_batches.exe**, we execute them by double click. First you need to accept the agreements.



The Batch Modules are **installed** to your **Desktop** by default. Click on **Browse** button if you want to **choose an other destination**:



After clicking **Install** the **Batch Modules** are **extracted**. So after that we will have a **new folder** on our **Desktop** called **VU_Batchs**.

Executing PrepLab.cmd

Like all WAIK cmd tools the **VU Batch Modules** have to run with **full administrative privileges**.

Important note for 64 bit OS users:

You need to **adjust** the value of **ImgX** variable BEFORE executing the **PrepLab.cmd**:

```
"%Programfiles%\Windows AIK\Tools\x86"
```

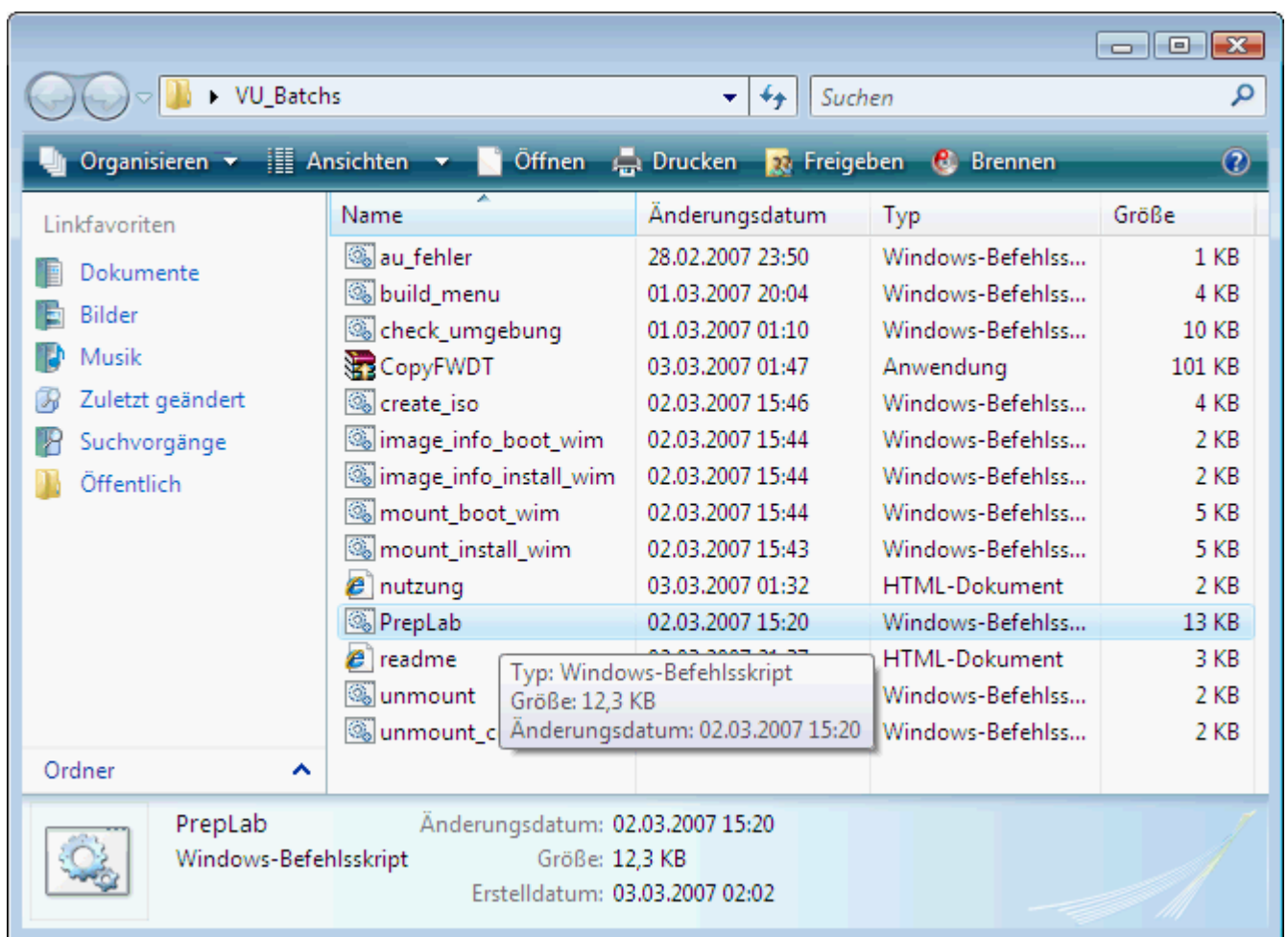
instead of "x86" you need to use "amd64" at the end of the line so it would look like this:

```
"%Programfiles%\Windows AIK\Tools\amd64"
```

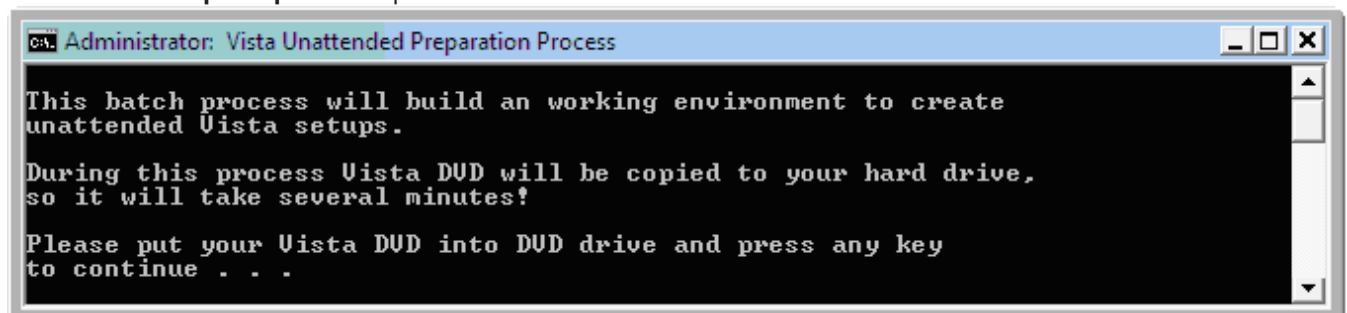
so the whole line looks like this:

```
SETX ImgX "%Programfiles%\Windows AIK\Tools\amd64" -m
```

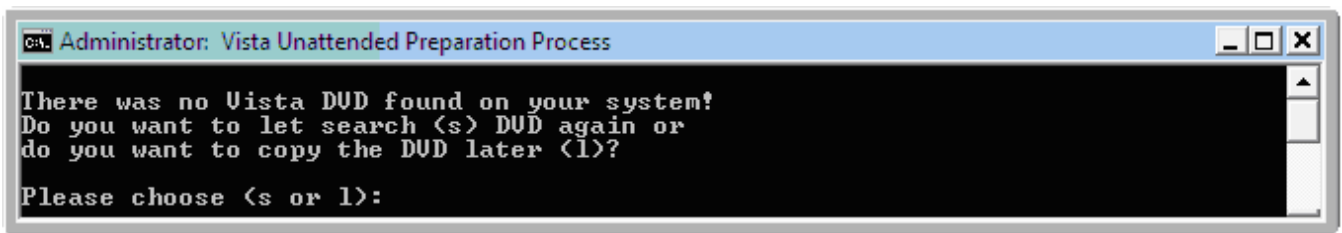
We **open** this new **folder** and **execute** the **PrepLab.cmd** by **double clicking**:



The first screen **prompts** us to put the **Vista DVD** into DVD drive:



It's **not a must** to copy the DVD at this point but I **recommend** doing so. If you want to skip copying DVD at this point, just hit a key without DVD inside the DVD drive and you will get the following prompt:

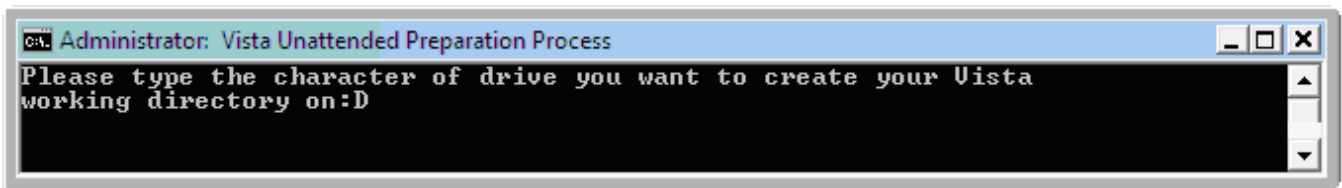


```
Administrator: Vista Unattended Preparation Process

There was no Vista DVD found on your system!
Do you want to let search <s> DVD again or
do you want to copy the DVD later <l>?

Please choose <s or l>:
```

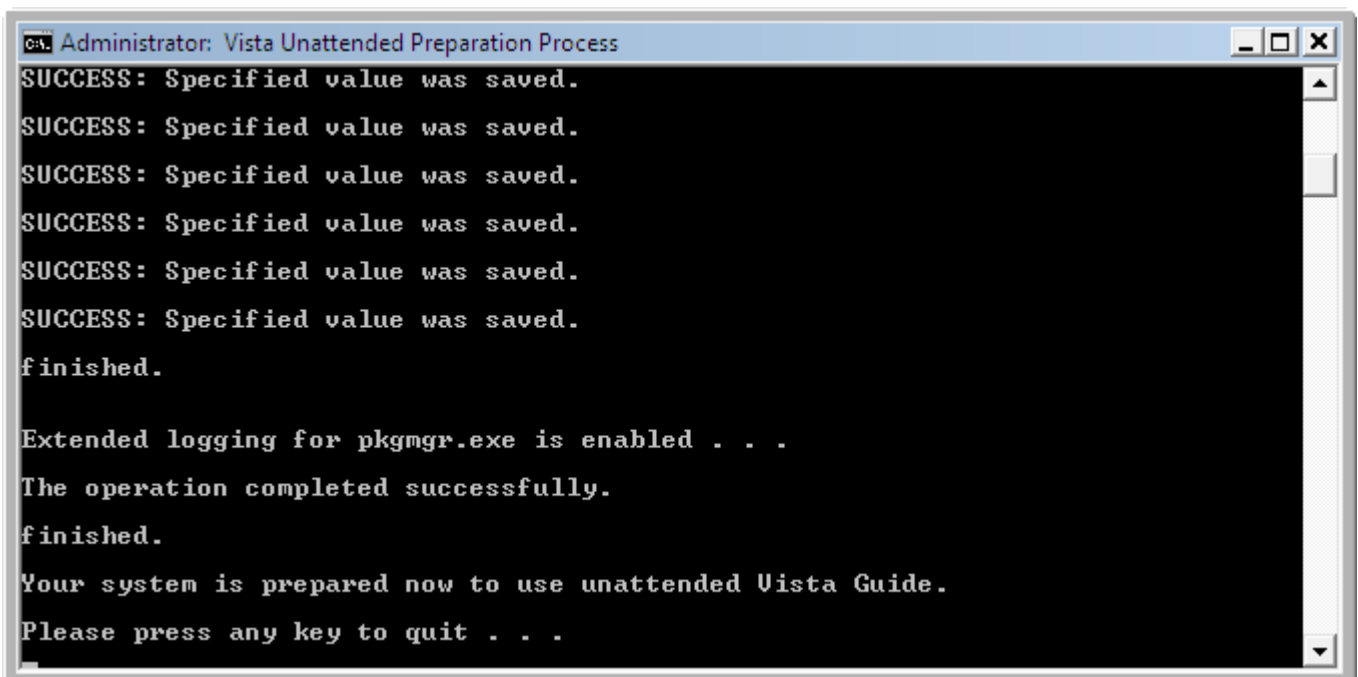
We choose **s or l** and we will be prompted to enter a **drive letter** of the drive we want the **working directory** create on. We ensure that we **enough space** left on the drive (> **6 GB recommended**) and enter the drive letter:



```
Administrator: Vista Unattended Preparation Process

Please type the character of drive you want to create your Vista
working directory on:D
```

During next step the working directory structure will be created on the drive we entered before and environment variables are settled to access the working directories:



```
Administrator: Vista Unattended Preparation Process

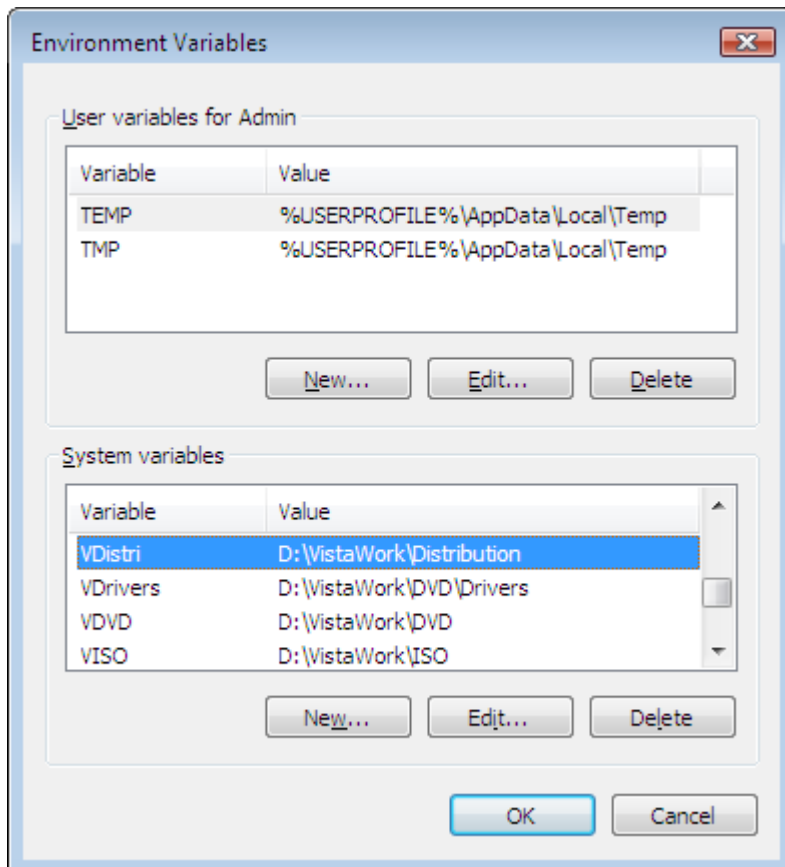
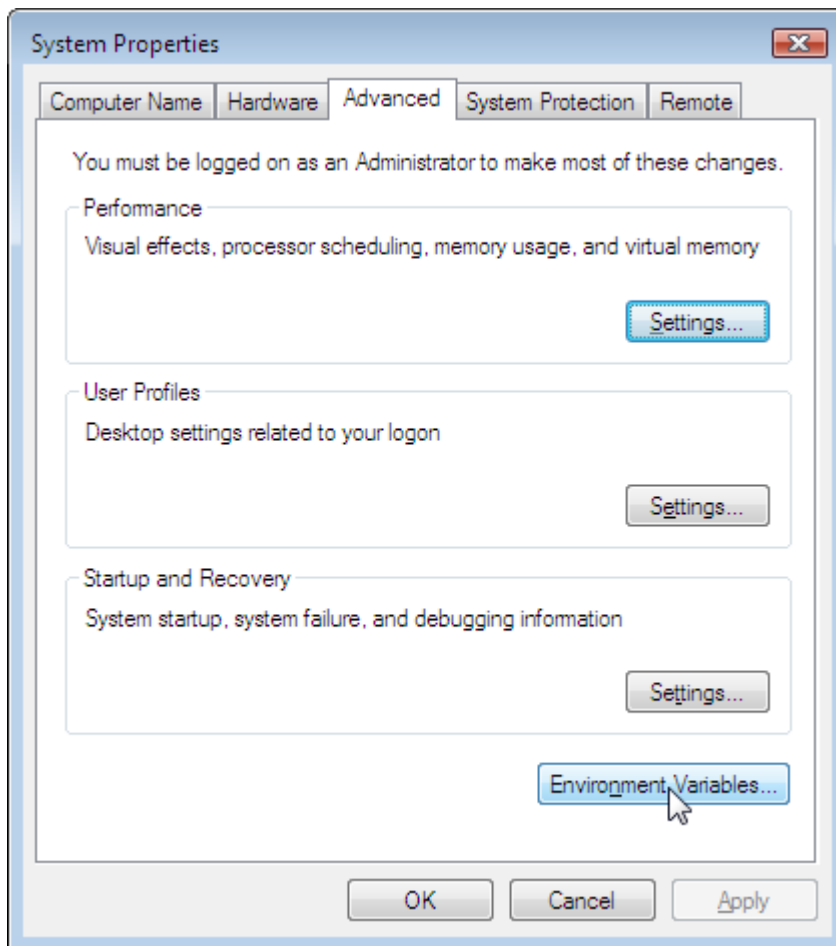
SUCCESS: Specified value was saved.
SUCCESS: Specified value was saved.
SUCCESS: Specified value was saved.
SUCCESS: Specified value was saved.
SUCCESS: Specified value was saved.
SUCCESS: Specified value was saved.
finished.

Extended logging for pkgmgr.exe is enabled . . .
The operation completed successfully.
finished.

Your system is prepared now to use unattended Vista Guide.
Please press any key to quit . . .
```

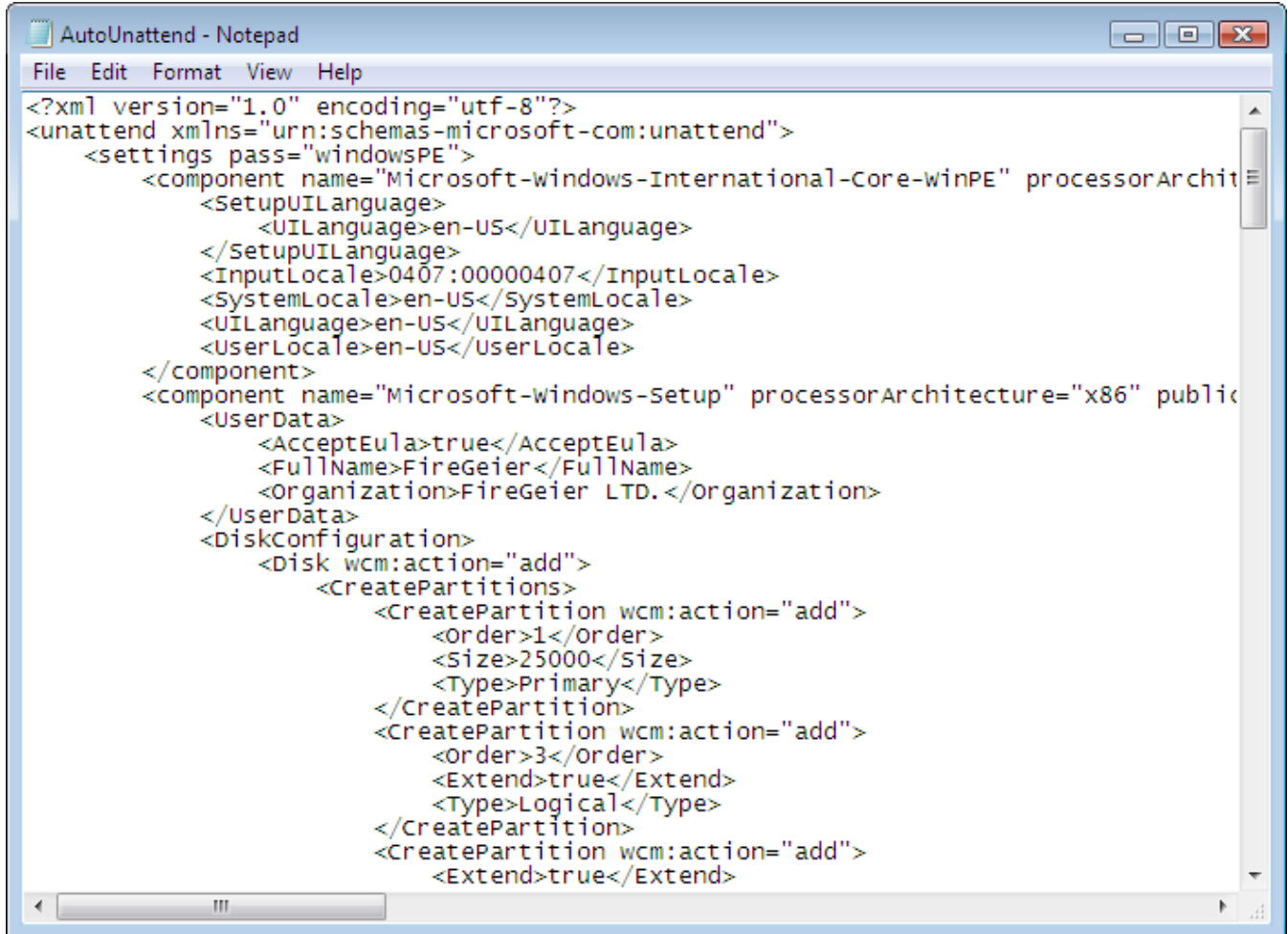
Now our **system is prepared** to execute the other VU Batch Modules and we can close the PrepLab.cmd by pressing any key.

If you want to find out which variables were set, click on **Start \ Control Panel \ System \ Advanced \ Environment Variables** to find out:



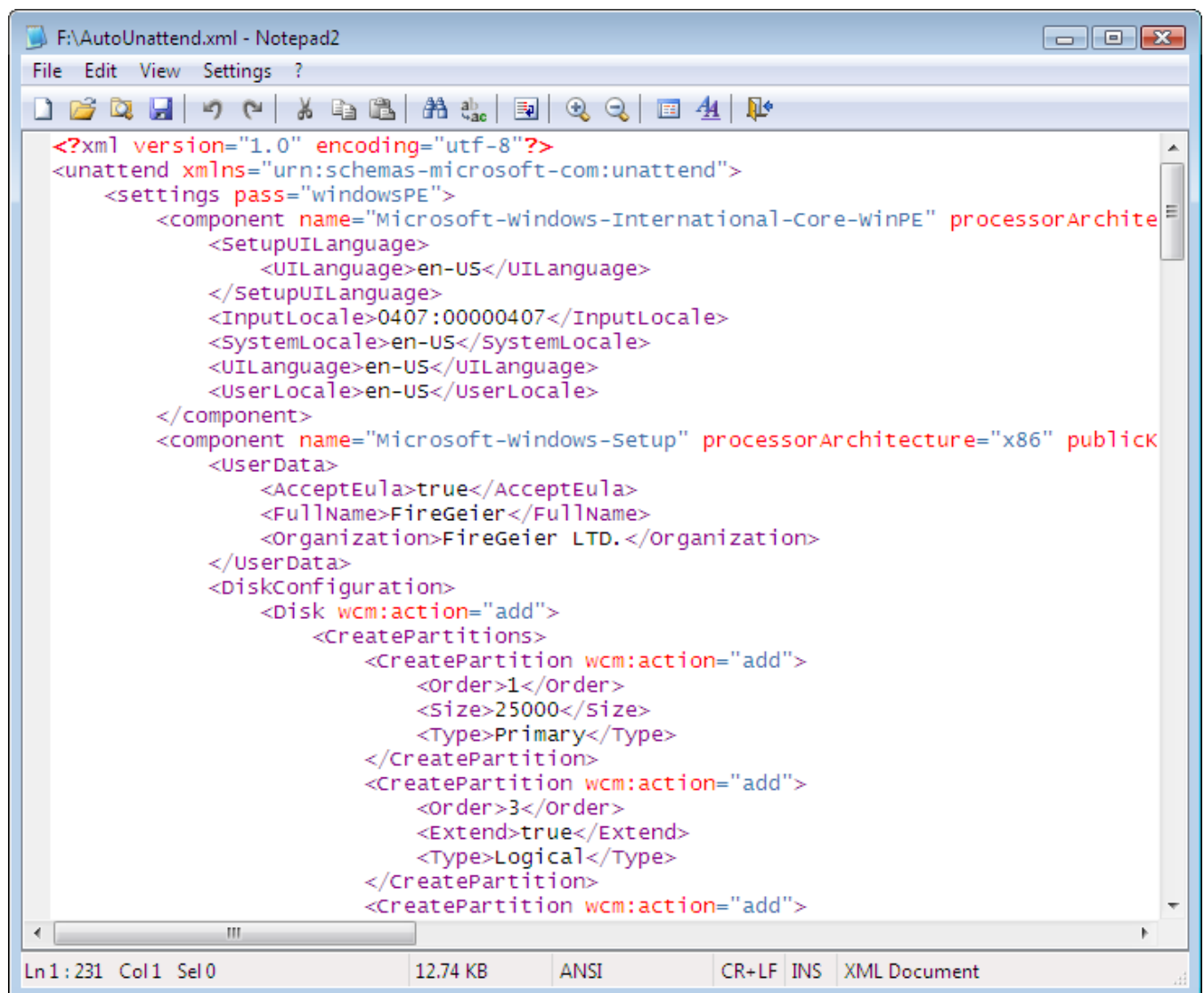
4. Tools we can install optional

We need to check our answer file by using any kind of editor and we may want to create our own batch files. You can do all that kind of work with Windows build in notepad.exe. But xml files are hard to read in notepad. Here is an example:



```
<?xml version="1.0" encoding="utf-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
  <settings pass="windowsPE">
    <component name="Microsoft-windows-International-Core-winPE" processorArchitecture="x86" publicKeyToken="31bf3856ad364e35" language="en-US" version="6.0.6002.18005" type="winPE" wcm:action="add">
      <SetupUILanguage>
        <UILanguage>en-US</UILanguage>
      </SetupUILanguage>
      <InputLocale>0407:00000407</InputLocale>
      <SystemLocale>en-US</SystemLocale>
      <UILanguage>en-US</UILanguage>
      <UserLocale>en-US</UserLocale>
    </component>
    <component name="Microsoft-windows-Setup" processorArchitecture="x86" publicKeyToken="31bf3856ad364e35" language="en-US" version="6.0.6002.18005" type="winPE" wcm:action="add">
      <UserData>
        <AcceptEula>true</AcceptEula>
        <FullName>FireGeier</FullName>
        <Organization>FireGeier LTD.</Organization>
      </UserData>
      <DiskConfiguration>
        <Disk wcm:action="add">
          <CreatePartitions>
            <CreatePartition wcm:action="add">
              <Order>1</Order>
              <Size>25000</Size>
              <Type>Primary</Type>
            </CreatePartition>
            <CreatePartition wcm:action="add">
              <Order>3</Order>
              <Extend>true</Extend>
              <Type>Logical</Type>
            </CreatePartition>
            <CreatePartition wcm:action="add">
              <Order>4</Order>
              <Extend>true</Extend>
              <Type>Logical</Type>
            </CreatePartition>
          </CreatePartitions>
        </Disk>
      </DiskConfiguration>
    </component>
  </settings>
</unattend>
```

A more powerful and handy free editor is [notepad2](#). There are others editors for free as well certainly. However here is the same example from above inside [notepad2](#):



```
<?xml version="1.0" encoding="utf-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
  <settings pass="windowsPE">
    <component name="Microsoft-windows-International-Core-winPE" processorArchitecture="x86">
      <SetupUILanguage>
        <UILanguage>en-US</UILanguage>
      </SetupUILanguage>
      <InputLocale>0407:00000407</InputLocale>
      <SystemLocale>en-US</SystemLocale>
      <UILanguage>en-US</UILanguage>
      <UserLocale>en-US</UserLocale>
    </component>
    <component name="Microsoft-windows-Setup" processorArchitecture="x86" publicKey="...">
      <UserData>
        <AcceptEula>true</AcceptEula>
        <FullName>FireGeier</FullName>
        <Organization>FireGeier LTD.</Organization>
      </UserData>
      <DiskConfiguration>
        <Disk wcm:action="add">
          <CreatePartitions>
            <CreatePartition wcm:action="add">
              <Order>1</Order>
              <Size>25000</Size>
              <Type>Primary</Type>
            </CreatePartition>
            <CreatePartition wcm:action="add">
              <Order>3</Order>
              <Extend>true</Extend>
              <Type>Logical</Type>
            </CreatePartition>
            <CreatePartition wcm:action="add">
              <Order>4</Order>
              <Extend>true</Extend>
              <Type>Logical</Type>
            </CreatePartition>
          </CreatePartitions>
        </Disk>
      </DiskConfiguration>
    </component>
  </settings>
</unattend>
```

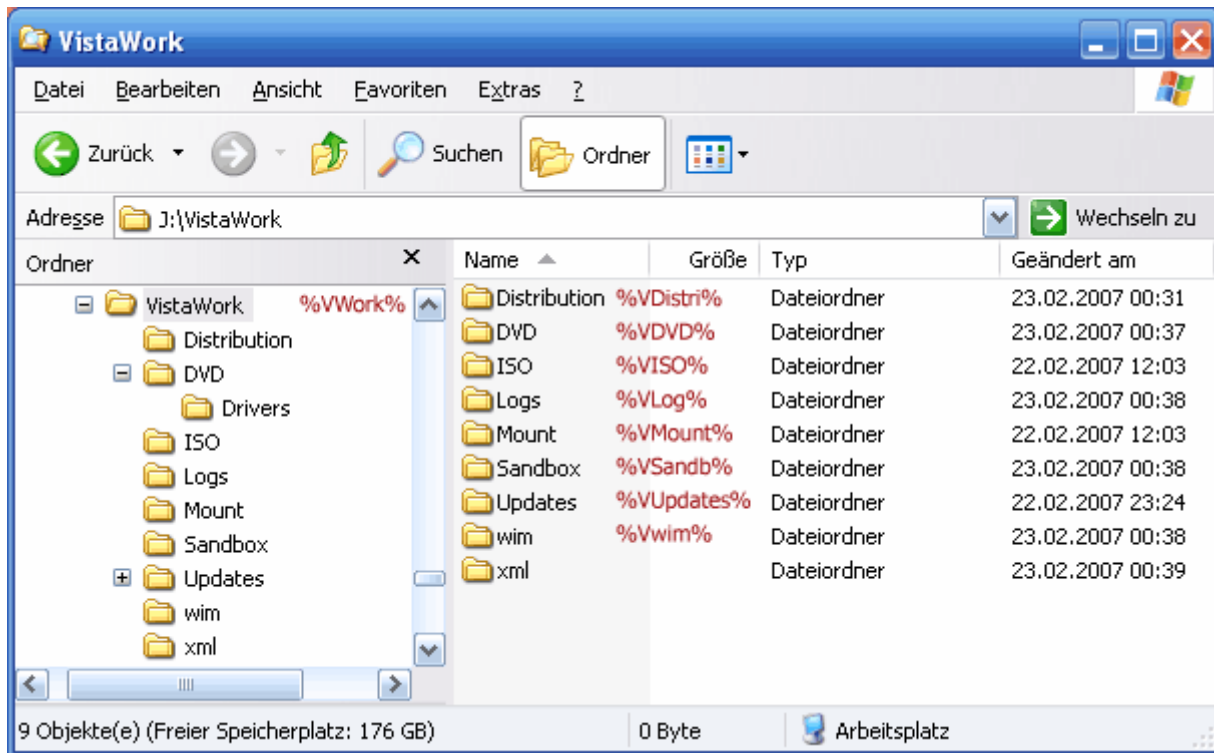
Ln 1: 231 Col 1 Sel 0 12.74 KB ANSI CR+LF INS XML Document

Build folder structure

1. Working directory structure

This section does explain the **folder structure** and **environment variables** which were set by **PrepLab.cmd**. If you don't like to run the PrepLab.cmd or if you've had problems executing it you can find out here how the structure is build. So you could do the same thing manually, if necessary.

After PrepLab.cmd has finished we will find a directory called **VistaWork** on the drive we've insert before in PrepLab.cmd. It contains the following folder structure:



The **red labels** behind the folder names are the **corresponding environment variables** pointing to the folder. So we can use these environment variables to access the corresponding folder from inside a batch or a command line.

2. Purpose of directories

Distribution (%VDistri%)

This folder does contain the **Distribution Share** folder structure.

DVD (%VDVD%)

This folder contains the **Vista DVD files** and **folders**. Furthermore there is a folder called **Drivers** inside. This folder will hold our additional drivers later.

ISO (%VISO%)

This folder contains the **ISO files** created for an unattended setup.

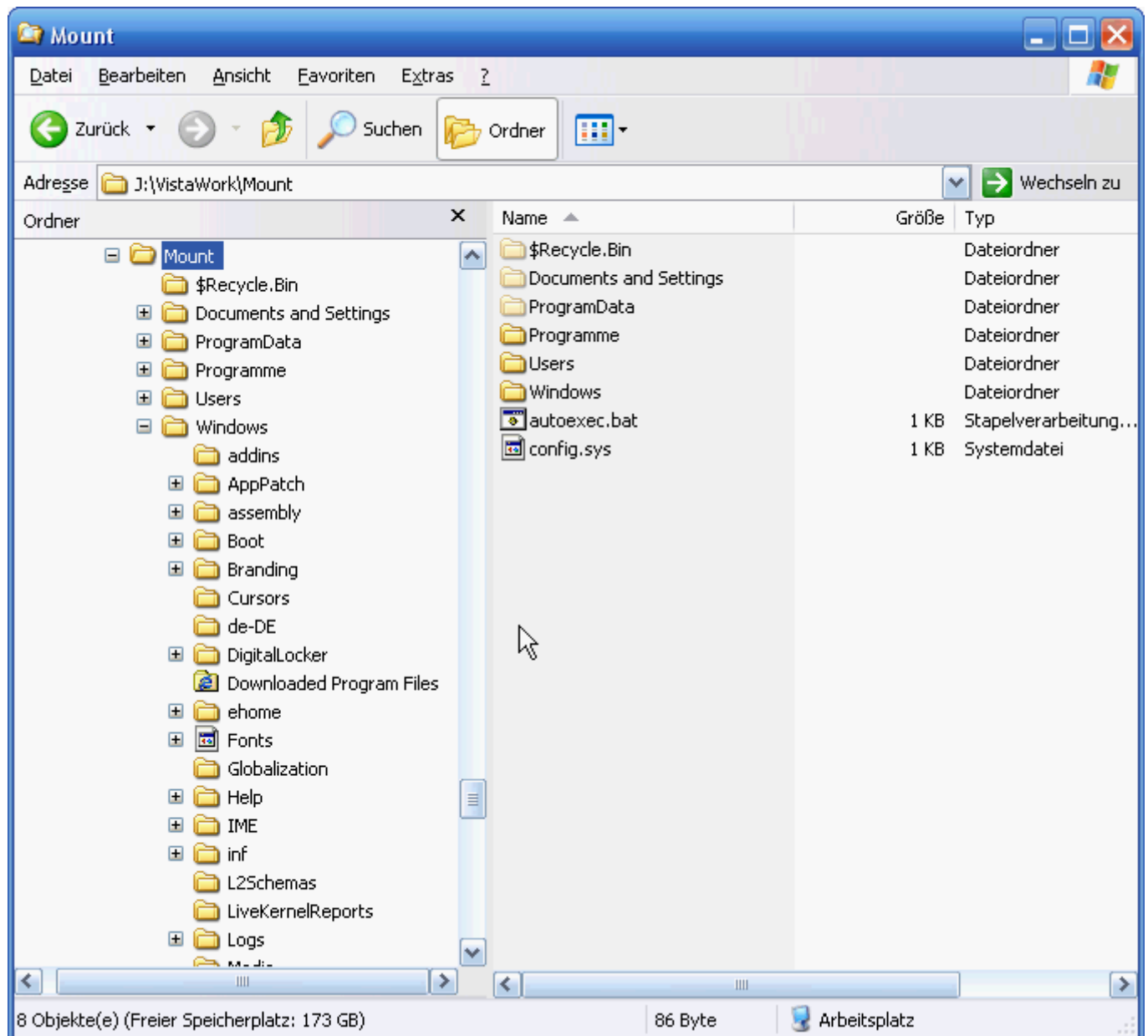
Logs (%VLog%)

Some of the WAIK command line tools will use this folder to store **log files**.

Mount (%VMount%)

Vista does store all setup files inside a packed image. So if we want to edit the files we need to mount the contents of the image to a directory first. So the Mount folder is used to **cache the mounted files for editing**.

As soon as we have mounted the image to the Mount folder, the structure inside the Mount folder will be the same like we will have later on our target system finally:



If we want to delete **already existing files or folders** from Mount directory, we need to press **Shift-Del**. **Take care** if you want to delete any existing files or folders. If you **delete** the **wrong ones** you will **may not be able to use the image** for setup any more!

Sandbox (%VSandb%)

The Sandbox folder is used by **package manager** as a **temporary working directory**, while injecting additional packages - like updates for example - to Vista image file. The packages will be extracted to the Sandbox folder than, dependencies will be checked and then they will be integrated into the mounted image.

Important:

After a package manager run we **MUST delete all files inside Sandbox folder**. Otherwise it could cause conflicts during the next package manager run!

Updates (%VUpdates%)

Here we will store all **MSU files** downloaded from Microsoft website.

wim (%Vwim%)

We can use this folder to store self build **Vista image files (.wim)**.


xml

This folder is used to store xml answer files we may not use for the actual unattended setup.

VU Batch Modules

How to use VU Batch Modules

If you **want to execute** one of the batch files (cmd), you just need to **double click** it with the **left** mouse button.

Inside the **guide** a little **batch symbol**  will **indicate** that you can do the following **step using one of the batch files**. Just move the **mouse over** that symbol and you will see a **hint**, telling you which cmd you need to execute:



If you **click** on the **batch symbol** you will be **forwarded to** the **cmd** on this site.

NOTE:

Like all WAIK cmd tools the **VU Batch Modules** have to run with **full administrative privileges**.

The batch files

This is an overview about available batch modules. The batch files containing detailed comments so you will read a short description of purpose here only.

I want to explicit point out to, that the batch modules are not like a final software program. They have just the purpose to avoid a lot of typing and they are an example how to automate processes of building unattended setups.

By modular structure of the batch files and by use of collective working environment, it's easy to create own batches.

Furthermore new batch files will be published on this site in future, too.

PrepLab.cmd

Purpose :

- Building working directory structure
- Copying Vista DVD to \VistaWork\DVD
- Setting environment variables

Important note for 64 bit OS users:

You need to **adjust** the value of **ImgX** variable BEFORE executing the **PrepLab.cmd**:

"%Programfiles%\Windows AIK\Tools\x86"

instead of "x86" you need to **use "amd64"** at the end of the line so it would look like this:

"%Programfiles%\Windows AIK\Tools\amd64"

so the whole line looks like this:

SETX Imgx "%Programfiles%\Windows AIK\Tools\amd64" -m

```

@ECHO OFF
TITLE Vista Unattended Preparation Process

:-----
::Purpose: Preparing System to use with cmd tools easier
::
::CALL: Called by user
::
::Version: 1.0
:-----

::
:: _____
::  MAIN PROCEDURE
:: _____

:Start
CLS
SET Label=
SET Device=
ECHO.
ECHO This batch process will build an working environment to create
ECHO unattended Vista setups.
ECHO.
ECHO During this process Vista DVD will be copied to your hard drive,
ECHO so it will take several minutes!
ECHO.
ECHO Please put your Vista DVD into DVD drive and press any key
ECHO to continue . . .
PAUSE > NUL

::Sets the drive letter of the DVD drive and stores it to DVDDrive variable.
::The Vista DVD needs to be inside drive find out.
:SetDVD
SET DVDDrive=NULL
FOR %%i IN (C D E F G H I J K L M N O P Q R S T U V W X Y Z) DO IF EXIST %%i:\sources\install.wim
SET DVDDrive=%%i:\
IF "%DVDDrive%"=="NULL" CALL :NoDVDDrive
SET Copy=true
IF "%DVDDrive%"=="NULL" SET Copy=false

::Asks where the working directory supposed to be created.
:Laufwerk
CLS
SET Label=Laufwerk
SET Device=Laufwerk
SET WorkDrive=
ECHO Please type the character of drive you want to create your Vista
SET /P WorkDrive= working directory on:
SET WorkDrive=%WorkDrive:~0,1%
CALL :CheckDrive %WorkDrive% %Device% %Label%

::All necessary working driectories are created using mkdir command.
:Verzeichnisse
CLS
ECHO.
ECHO Creating driectories . . .
IF NOT EXIST %WorkDrive%\VistaWork\NUL MKDIR %WorkDrive%\VistaWork
IF NOT EXIST %WorkDrive%\VistaWork\Deploy\NUL MKDIR %WorkDrive%\VistaWork\Deploy
IF NOT EXIST %WorkDrive%\VistaWork\DVD\NUL MKDIR %WorkDrive%\VistaWork\DVD
IF NOT EXIST %WorkDrive%\VistaWork\DVD\Drivers\NUL MKDIR %WorkDrive%\VistaWork\DVD\Drivers
IF NOT EXIST %WorkDrive%\VistaWork\ISO\NUL MKDIR %WorkDrive%\VistaWork\ISO
IF NOT EXIST %WorkDrive%\VistaWork\Logs\NUL MKDIR %WorkDrive%\VistaWork\Logs
IF NOT EXIST %WorkDrive%\VistaWork\Mount\NUL MKDIR %WorkDrive%\VistaWork\Mount
IF NOT EXIST %WorkDrive%\VistaWork\Distribution\NUL MKDIR %WorkDrive%\VistaWork\Distribution
IF NOT EXIST %WorkDrive%\VistaWork\wim\NUL MKDIR %WorkDrive%\VistaWork\wim
IF NOT EXIST %WorkDrive%\VistaWork\Sandbox\NUL MKDIR %WorkDrive%\VistaWork\Sandbox
IF NOT EXIST %WorkDrive%\VistaWork\Updates\NUL MKDIR %WorkDrive%\VistaWork\Updates
IF NOT EXIST %WorkDrive%\VistaWork\xml\NUL MKDIR %WorkDrive%\VistaWork\xml
ECHO.
ECHO finished.

::The Vista DVD is copied to %WorkDrive%\VistaWork\DVD\ on HD using XCOPY. "/cheriy" are
::the parameters used with XCOPY (start cmd.exe and type "xcopy /?" to find out more about).
::
::START /WAIT ensures, that batch is executed only once copying procedure is finished.
::/min means, that the copying process is started inside minimized window.

```

```

::
::"Copying Vista DVD..." is the title of the process window. The title is optional usually
::but MUST be setted, if the following call is in quotes cause including spaces.
::If you don't use a title the call will be interpreted as windows title and the process
::will fail!!!

::Fist of all user will be aksed, if he wants to copy the DVD to HD or not.

IF "%Copy%"=="true" GOTO DVDKopieren
IF "%Copy%"=="false" GOTO NichtKopieren

:DVDKopieren
ECHO.
ECHO Vista-DVD will be copied to working directory . . .
START /WAIT /min "Copying Vista DVD..." XCOPY %DVDDrive%*. * %WorkDrive%\VistaWork\DVD\ /cheriy
ECHO.
ECHO finished.
GOTO CheckPkg

::If DVD is not copied, user will get a message where he has to copy it to later.

:NichtKopieren
ECHO.
ECHO -----
ECHO PLEASE NOTE:
ECHO Cause no DVD will be copied now, you need to copy Vista DVD
ECHO later to the following directory:
ECHO "%WorkDrive%\VistaWork\DVD"
ECHO -----
ECHO.

::Checks, if pkgmgr.exe is present inside default directory.

:CheckPkg
IF NOT EXIST "%Programfiles%\Windows AIK\Tools\Servicing\pkgmgr.exe" GOTO NoPkg

::Using SETX you can define environment variables during running windows session.
::The variable are writte to user environment by default.
::
::-m option will set the variables to system environment.
::The syntax looks like this:
::SETX -m name value

:Umgebung
ECHO.
IF NOT EXIST %WINDIR%\System32\setx.exe (Goto ManuellSet)
ECHO Environment variables are setted . . .
SETX VWork %WorkDrive%\VistaWork -m
SETX VDVD %WorkDrive%\VistaWork\DVD -m
SETX VDrivers %WorkDrive%\VistaWork\DVD\Drivers -m
SETX VDistri %WorkDrive%\VistaWork\Distribution -m
SETX VISO %WorkDrive%\VistaWork\ISO -m
SETX VLog %WorkDrive%\VistaWork\Logs -m
SETX VMount %WorkDrive%\VistaWork\Mount -m
SETX Vwim %WorkDrive%\VistaWork\wim -m
SETX VUpdates %WorkDrive%\VistaWork\Updates -m
SETX VSandb %WorkDrive%\VistaWork\Sandbox -m

:*****
::IMPORTANT NOTE FOR 64 BIT OS USERS:
:*****
::You need to adjust the value of ImgX variable BEFORE executing the PrepLab.cmd:
::
::"%Programfiles%\Windows AIK\Tools\x86"
::
::instead of "x86" you need to use "amd64" at the end of the line so it would look
::like this:
::
::"%Programfiles%\Windows AIK\Tools\amd64"
::
::so the whole line looks like this:
::
::SETX ImgX "%Programfiles%\Windows AIK\Tools\amd64" -m
SETX ImgX "%Programfiles%\Windows AIK\Tools\x86" -m

```

```

SETX PETools "%Programfiles%\Windows AIK\Tools\PETools" -m
IF NOT "%PkgMgr%"=="false" SETX PMgrDir "%Programfiles%\Windows AIK\Tools\Servicing" -m
IF EXIST "%WorkDrive%\VistaWork\DVD\boot\etfsboot.com" SETX VVersion x86 -m
IF EXIST "%WorkDrive%\VistaWork\DVD\boot\cdboot.efi" SETX VVersion amd64 -m
ECHO.
ECHO finished.

::Set extend device install log options for pkgmgr.exe
::This will set a Regkey that's necessary to create a log while injecting drivers into
::install.wim
ECHO.
ECHO.
ECHO Extended logging for pkgmgr.exe is enabled . . .
ECHO.
reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows\CurrentVersion\Device Installer" /v DebugPkgMgr /t
REG_DWORD /d 00000001 /f
ECHO.
ECHO finished.

ECHO.
ECHO Your system is prepared now to use unattended Vista Guide.
ECHO.
ECHO Please press any key to quit . . .
PAUSE > NUL
EXIT

::
:: SUB PROCEDURES & PARTS EXECUTED OPTIONAL
::

::This section is executed if setx.exe was not found on system

:ManuellSet
ECHO.
ECHO SETX.EXE was not found on your system. So it's not possible
ECHO to set environment variables automatically on your system.
ECHO.
ECHO Please set the following environment variables manually
ECHO using Control Panal. Set them to System environment:
ECHO.
ECHO Name          Wert
ECHO -----
ECHO VWork          %WorkDrive%\VistaWork
ECHO VDVD           %WorkDrive%\VistaWork\DVD
ECHO VDrivers       %WorkDrive%\VistaWork\DVD\Drivers
ECHO VISO           %WorkDrive%\VistaWork\ISO
ECHO VLogs          %WorkDrive%\VistaWork\Logs
ECHO Mount          %WorkDrive%\VistaWork\Mount
ECHO VImage         %WorkDrive%\VistaWork\Image
ECHO VUpdates       %WorkDrive%\VistaWork\Updates
ECHO VSandb         %WorkDrive%\VistaWork\Sandbox
ECHO PMgrDir        %Programfiles%\Windows AIK\Tools\Servicing
ECHO ImgX           %Programfiles%\Windows AIK\Tools\x86
ECHO PETools        %Programfiles%\Windows AIK\Tools\PETools
ECHO.
ECHO VVersion       x86                      (fuer Vista 32bit)
ECHO ODER
ECHO VVersio        amd64                    (fuer Vista 64bit)
ECHO -----
ECHO.
ECHO You can reach these properties under
ECHO.
ECHO Control Panal\System\Advanced\Environment Variables
ECHO.
ECHO Please set them to system environment!
ECHO.
ECHO Press any key to quit . . .
PAUSE > NUL
EXIT

::Executed if package manager (pkgmgr.exe) is not located inside
::%Programfiles%\Windows AIK\Tools\Servicing\pkgmgr.exe

:NoPkg
Set PkgMgr=false
ECHO.
ECHO The package manager (pkgmgr.exe) was not found inside

```

```

ECHO %Programfiles%\Windows AIK\Tools\Serviceing
ECHO.
ECHO Please ensure that WAIK was installed properly on your system.
ECHO.
ECHO You may need to go to
ECHO Control Panal\System\Advanced\Environment Variables
ECHO and set following environment variables manually.
ECHO.
ECHO Name                      Wert
ECHO -----
ECHO PMgrDir                    PfadzurPKGMGR.exe
ECHO.
ECHO Beispiel:
ECHO PMgrDir                    C:\Programme\WAIK\Tools
ECHO.
ECHO.
ECHO Please press any key to continue . . .
PAUSE > NUL
GOTO Umgebung

::NoDVDDrive is called from SetDVD section, if DVDDrive is still NULL. That means no DVD
::was found.
::
::A message is displayed and user can choose, if he wants to look for the DVD again or if
::he wants to skip the copy procedure for the moment.
::
::Depending on user choice SetDVD section is executed again (GOTO SetDVD)
::or process is continued (GOTO :EOF) without copying DVD for the moment.
::
::If user choose neather "s" nor "l" NoDVDDrive section is repeated.

:NoDVDDrive
CLS
ECHO.
ECHO There was no Vista DVD found on your system!
ECHO Do you want to let search (s) DVD again or
ECHO do you want to copy the DVD later (l)?
ECHO.
SET /P Wahl=Please choose (s or l):
IF "%Wahl%"=="s" (GOTO SetDVD) ELSE IF "%Wahl%"=="l" (GOTO :EOF)
GOTO NoDVDDrive

::Section checks, if the device assigned by CALL :CheckDrive does exist at all.
::%1, %2, %3 are equivalent to the variables inside CALL command.
::Here they are: %1=%WorkDrive%, %2=%Device% und %3=%Label%

:CheckDrive
IF EXIST %1\Nul GOTO :EOF
ECHO.
ECHO The %2 %1 is NOT valid!
ECHO.
ECHO Please press any key to continue . . .
PAUSE > NUL
GOTO %3

```

image_info_boot_wim.cmd and image_info_install_wim.cmd

Purpose:

- Shows informations about .wim files (Number of contained images, names of images etc.)

The **image_info_boot_wim.cmd** and the **image_info_install_wim.cmd** are **exactly the same** cmd files, **except** the value of **%Abbild%** variable. The %Abbild% variable points to the .wim file you want to have the information about. So you could create further info.cmd files for your own .wim files just by editing the value of %Abbild% variable.

```
@ECHO OFF
TITLE Image Informations

:-----
::Purpose: Batch will collect index information about boot.wim
::
::CALL: Called by user
::
::Version: 1.0
:-----

::Check environment
::Fist of all is checked, if all used environment variables are declared and valid and if
::all used directories do exist. This done by check_umgebung.cmd.
::
::check_umgebung.cmd will assign a value "true" to EnvErr variable, if there is an error in
::working environment and it will assign "false" to EnvErr variable, if everything is alright
::in working evrionment.
::
::If there is an error in working environment (EnvErr=true) au_fehler.cmd is called.
::au_fehler.cmd will display an error message and quit the batch.

CALL check_umgebung.cmd
IF %EnvErr%==true CALL au_fehler.cmd

:-----
::
::  MAIN PROCEDURE
:-----

::DECLARING Variables
::You need to adpat the following SET lines, if your system was not prepared or not prepared
::correctly using preplab.cmd. Remove everything in front of SET and adjust the path behind
::"=". Expect ImgX - this is path where imagex.exe is located - all pahtes are arbitrary.
::
::The Abbild variable needs to be adjusted depending on which image (.wim file) you want
::to use for the operation. It's recommended to save the .cmd with an other name, if you
::want to use it for another .wim file. So you don't need to adjust it every time you want
::to use it with the other .wim file

:----- Start Adjustment-----
SET Abbild=%VDVD%\sources\boot.wim
::SET ImgX="%Programfiles%\Windows AIK\Tools\x86"
::SET VDVD=D:\VistaWork\DVD
:-----End Adjustment-----

::%VDVD% was declared by preplab.cmd and is x:\VistaWork\DVD. x represants the drive letter
::which was insert during preplab.cmd run.
::
::imagex /info lists the informations about the insert .wim file - here of boot.wim.
::That are informations like number and size of contained images, their name etc.

ECHO.
ECHO Collecting Image informations . . .
ECHO.
"%ImgX%\imagex /info %Abbild%
ECHO.
ECHO finished.
ECHO.

ECHO Please press any key to quit . . .
PAUSE >NUL
EXIT
```

```

@ECHO OFF
TITLE Image Informations

:-----
::Purpose: Batch will collect index information about install.wim
::
::CALL: Called by user
::
::Version: 1.0
:-----

::Check environment
::Fist of all is checked, if all used environment variables are declared and valid and if
::all used directories do exist. This done by check_umgebung.cmd.
::
::check_umgebung.cmd will assign a value "true" to EnvErr variable, if there is an error in
::working environment and it will assign "false" to EnvErr variable, if everything is alright
::in working evrionment.
::
::If there is an error in working environment (EnvErr=true) au_fehler.cmd is called.
::au_fehler.cmd will display an error message and quit the batch.

CALL check_umgebung.cmd
IF %EnvErr%==true CALL au_fehler.cmd

::DECLARING Variables
::You need to adpat the following SET lines, if your system was not prepared or not prepared
::correctly using preplab.cmd. Remove everything in front of SET and adjust the path behind
::"=". Expect ImgX - this is path where imagex.exe is located - all pahtes are arbitrary.
:----- Start Adjustment-----
SET Abbild=%VDVD%\sources\install.wim
::SET ImgX="%Programfiles%\Windows AIK\Tools\x86"
::SET VMount=D:\VistaWork\Mount
:-----End Adjustment-----

::%VDVD% was declared by preplab.cmd and is x:\VistaWork\DVD. x represants the drive letter
::which was insert during preplab.cmd run.
::
::imagex /info lists the informations about the insert .wim file - here of install.wim.
::That are informations like number and size of contained images, their name etc.

ECHO.
ECHO Collecting Image informations . . .
ECHO.
"%ImgX%\imagex /info %Abbild%
ECHO.
ECHO abgeschlossen.
ECHO.

ECHO Please press any key to quit . . .
PAUSE >NUL
EXIT

```

mount_boot_wim.cmd and mount_install_wim.cmd

Purpose:

- A .wim file declared by %Abbild% variable is **mounted** to \VistaWork\Mount for **reading and writing** by using **imagex /mountrw** command.
- If an **image is mounted already** to \VistaWork\Mount, the batch offers the **option to close the mounted image** before mounting the new one.

mount_boot_wim.cmd and **mount_install_wim.cmd** are exactly the same batch files again, expect the value of %Abbild% variable. So you could create further mount .cmd files for your own .wim files just by editing the value of %Abbild% variable.

```
@ECHO OFF
TITLE Mounting Image . . .

:-----
::Purpose: Batch will mount boot.wim to %VMount%
::
::CALL: From other batchs or user
::
::Version: 1.0
:-----

::Check environment
::Fist of all is checked, if all used environment variables are declared and valid and if
::all used directories do exist. This done by check_umgebung.cmd.
::
::check_umgebung.cmd will assign a value "true" to EnvErr variable, if there is an error in
::working environment and it will assign "false" to EnvErr variable, if everything is alright
::in working evrionment.
::
::If there is an error in working environment (EnvErr=true) au_fehler.cmd is called.
::au_fehler.cmd will display an error message and quit the batch.

CALL check_umgebung.cmd
IF %EnvErr%==true CALL au_fehler.cmd

::DECLARING Variables
::You need to adapt the following SET lines, if your system was not prepared or not prepared
::correctly using preplab.cmd. Remove everything in front of SET and adjust the path behind
::"=". Expect ImgX - this is path where imagex.exe is located - all pahtes are arbitrary.
::
:----- Start Adjustment-----
SET Abbild=%VDVD%\sources\boot.wim
::SET ImgX="%Programfiles%\Windows AIK\Tools\x86"
::SET VMount=J:\VistaWork\Mount
::SET VLog=J:\VistaWork\Logs
::SET VDVD=J:\VistaWork\DVD
:-----End Adjustment-----

:-----
::
::      MAIN PROCEDURE
:-----

::Fist of all we need to create a menu of the different images contained inside .wim file.
::
::We can get the necessary informations using imagex.exe /info. To make the informations
::accessible from inside a batch we redirect the imagex output to index.txt file located
::in %VLog% (setted by perplab.cmd).
::
::If there would be an image mounted already, imagex will set %Errorlevel% system variable
::to 2. MountVoll subroutine will be called to handle that error.

:Anfang
CLS
"%ImgX%\imagex.exe /info %Abbild% > %VLog%\index.txt
IF "%Errorlevel%"=="2" GOTO MountVoll

CALL build_menu.cmd %Abbild%

::Next we need to find out which image is choosen. The value of choosen image is stored in
::Menu variable.
::
::First Index variable is set to NULL. Than a FOR loop will check, if one of the possible
::selcetions matchs the value of Menu variable. The loop starts with 1 and ends with value
```

```

::of ICount variable. The value of ICount depends on the number of images contained inside
::the .wim file.
::
::If the value of Menu variable does match the value of %%i variable the value will be
::set for Index variable.
::
::Only if Index variable will have an other value than NULL - means that there was a valid
::selection - the batch will continue. Otherwise Anfang section will be repeated.

ECHO.
SET /P Menu=Please choose an image (1-%%ICount%):
ECHO.
SET Index=NULL
FOR /L %%i IN (1,1,%%ICount%) DO IF %%Menu%==%%i (SET Index=%%i)
IF %%Index%==NULL GOTO Anfang

ECHO You have selected image %%Index%.

::The .wim file defined in Abbild variable is mounted for read and write operations
::using imagex /mountrw command to %VMount% directory.
::It will open the image selected before from menu (Index variable) thereby.

:MountLeer
ECHO.
ECHO Image with index %%Index% is mounted for editing to %VMount%.
START "ImageX" /WAIT "%ImgX%\imagex /mountrw %%Abbild% %%Index% %VMount%"
ECHO.
ECHO The image file was mounted. Please press any key to quit . . .
GOTO Ende

::
:: _____
::      SUB PROCEDURES & PARTS EXECUTED OPTIONAL
:: _____

::If an image is mounted already, user will be asked what to do now.
::
::User can choose, if he wants to cancel the whole procedure (abort), if he wants to
::unmount the image and save changes or if he wants to unmount the image without saving
::changes.

:MountVoll
CLS
ECHO.
ECHO There is an image mounted to %VMount% already.
ECHO.
ECHO Do you want to unmount and save the image (s)?
ECHO Du you want to unmount the image without saving (v)?
ECHO Do you want to abort (a)?
ECHO.
SET /P Choice=Please choose (s, v oder a):
IF "%Choice%"=="s" GOTO Speichern
IF "%Choice%"=="v" GOTO Verwerfen
IF "%Choice%"=="a" EXIT
GOTO MountVoll

::If user wants to unmount and save changes the unmount_commit.cmd is called. After that
::Anfang section is executed again.

:Speichern
CALL unmount_commit.cmd
GOTO Anfang

::If user wants to unmount the image only without any changes the unmount.cmd is called.
::After that Anfang section is executed again.

:Verwerfen
CALL unmount.cmd
GOTO Anfang

:Ende
PAUSE > NUL
EXIT

```

```

@ECHO OFF
TITLE Mounting Image . . .

:-----
::Purpose: Batch will mount install.wim to %VMount%
::
::CALL: From other batchs or user
::
::Version: 1.0
:-----

::Check environment
::First of all is checked, if all used environment variables are declared and valid and if
::all used directories do exist. This done by check_umgebung.cmd.
::
::check_umgebung.cmd will assign a value "true" to EnvErr variable, if there is an error in
::working environment and it will assign "false" to EnvErr variable, if everything is alright
::in working environment.
::
::If there is an error in working environment (EnvErr=true) au_fehler.cmd is called.
::au_fehler.cmd will display an error message and quit the batch.

CALL check_umgebung.cmd
IF %EnvErr%==true CALL au_fehler.cmd

::DECLARING Variables
::You need to adapt the following SET lines, if your system was not prepared or not prepared
::correctly using preplab.cmd. Remove everything in front of SET and adjust the path behind
::"=". Expect ImgX - this is path where imagex.exe is located - all paths are arbitrary.
::
::The Abbild variable needs to be adjusted depending on which image (.wim file) you want
::to use for the operation. It's recommended to save the .cmd with an other name, if you
::want to use it for another .wim file. So you don't need to adjust it every time you want
::to use it with the other .wim file
::
::----- Start Adjustment-----
SET Abbild=%VDVD%\sources\install.wim
::SET ImgX="%Programfiles%\Windows AIK\Tools\x86"
::SET VMount=J:\VistaWork\Mount
::SET VLog=J:\VistaWork\Logs
::SET VDVD=J:\VistaWork\DVD
::-----End Adjustment-----

:-----
::
::      MAIN PROCEDURE
:-----

::First of all we need to create a menu of the different images contained inside .wim file.
::
::We can get the necessary informations using imagex.exe /info. To make the informations
::accessible from inside a batch we redirect the imagex output to index.txt file located
::in %VLog% (setted by preplab.cmd).
::
::If there would be an image mounted already, imagex will set %Errorlevel% system variable
::to 2. MountVoll subroutine will be called to handle that error.

:Anfang
CLS
"%ImgX%\imagex.exe /info %Abbild% > %VLog%\index.txt
IF "%Errorlevel%"=="2" GOTO MountVoll

CALL build_menu.cmd %Abbild%

::Next we need to find out which image is chosen. The value of chosen image is stored in
::Menu variable.
::
::First Index variable is set to NULL. Then a FOR loop will check, if one of the possible
::selections matches the value of Menu variable. The loop starts with 1 and ends with value
::of ICount variable. The value of ICount depends on the number of images contained inside
::the .wim file.
::
::If the value of Menu variable does match the value of %i variable the value will be
::set for Index variable.
::
::Only if Index variable will have an other value than NULL - means that there was a valid

```

```

::selection - the batch will continue. Otherwise Anfang section will be repeated.

ECHO.
SET /P Menu=Please choose an image (1-%ICount%):
ECHO.
SET Index=NULL
FOR /L %%i IN (1,1,%ICount%) DO IF %Menu%==%%i (SET Index=%%i)
IF %Index%==NULL GOTO Anfang

ECHO You have selected image %Index%.

::The .wim file defined in Abbild variable is mounted for read and write operations
::using imagex /mountwim command to %VMount% directory.
::It will open the image selected before from menu (Index variable) thereby.

:MountLeer
ECHO.
ECHO The image %Index% of .wim file is mounted to %VMount% for editing.
START "ImageX" /WAIT "%ImgX%" \imagex /mountwim %Abbild% %Index% %VMount%
ECHO.
ECHO The image file was mounted. Please press any key to quit . . .
GOTO Ende

::
:: SUB PROCEDURES & PARTS EXECUTED OPTIONAL
::

::If an image is mounted already, user will be asked what to do now.
::
::User can choose, if he wants to cancel the whole procedure (abort), if he wants to
::unmount the image and save changes or if he wants to unmount the image without saving
::changes.

:MountVoll
CLS
ECHO.
ECHO There is an image mounted to %VMount% already.
ECHO.
ECHO Do you want to unmount and save the image (s)?
ECHO Do you want to unmount the image without saving (v)?
ECHO Do you want to abort (a)?
ECHO.
SET /P Choice=Please choose (s, v oder a):
IF "%Choice%"=="s" GOTO Speichern
IF "%Choice%"=="v" GOTO Verwerfen
IF "%Choice%"=="a" EXIT
GOTO MountVoll

::If user wants to unmount and save changes the unmount_commit.cmd is called. After that
::Anfang section is executed again.

:Speichern
CALL unmount_commit.cmd
GOTO Anfang

::If user wants to unmount the image only without any changes the unmount.cmd is called.
::After that Anfang section is executed again.

:Verwerfen
CALL unmount.cmd
GOTO Anfang

:Ende
PAUSE > NUL
EXIT

```

unmount.cmd

Purpose:

- The image mounted to **\VistaWork\Mount** is closed without saving any changes.

```
@ECHO OFF
TITLE Unmount without saving . . . (/unmount)

:-----
::Purpose: Unmounting an image located in %VMount%
::
::CALL: Called by user
::
::Version: 1.0
:-----

::DECLARING Variables
::You need to adapt the following SET lines, if your system was not prepared or not prepared
::correctly using preplab.cmd. Remove everything in front of SET and adjust the path behind
::"=". Expect ImgX - this is path where imagex.exe is located - all paths are arbitrary.
::---Anpassung Start-----
::SET ImgX="%Programfiles%\Windows AIK\Tools\x86"
::SET VMount=D:\VistaWork\Mount
::---Anpassung Ende-----

::
::  _____
::  MAIN PROCEDURE
::  _____

::%VMount% is the folder where the image file was mounted to.
::
::Imagex /unmount does "close" the image file. So content can't be edited/read any longer.

ECHO.
ECHO The Image will be unmounted without saving changes . . .
START "Imagex /unmount" /WAIT "%ImgX%\imagex /unmount %VMount%"
ECHO.
ECHO finished.
ECHO.
ECHO.
ECHO Image was unmounted. Please press any key to quit . . .
PAUSE >NUL
GOTO :EOF
```

unmount_commit.cmd

Purpose:

- The image mounted to **\VistaWork\Mount** is closed and all changes are saved.

```
@ECHO OFF
TITLE Unmounting Image saving changes (/unmount /commit)

:-----
::Purpose: Unmounting and saving an image located in %VMount%
::
::CALL: Called by user
::
::Version: 1.0
:-----

::DECLARING Variables
::You need to adapt the following SET lines, if your system was not prepared or not prepared
::correctly using preplab.cmd. Remove everything in front of SET and adjust the path behind
::"=". Expect ImgX - this is path where imagex.exe is located - all paths are arbitrary.
:----- Start Adjustment -----
::SET ImgX="%Programfiles%\Windows AIK\Tools\x86"
::SET VMount=D:\VistaWork\Mount
:----- End Adjustment -----

::%VMount% is the folder where the image file was mounted to.
::
::Imagex /unmount does "close" the image file. So content can't be edited/read any longer.
::
::/commit saves changes during unmounting image!

ECHO.
ECHO The image will be unmounted and changes will be saved . . .
START "Imagex /unmount /commit" /WAIT "%ImgX%\imagex /unmount /commit %VMount%"
ECHO.
ECHO finished.
ECHO.
ECHO.
ECHO The image was unmounted. Please press any key to quit . . .
PAUSE >NUL
GOTO :EOF
```

Function:

```

@ECHO OFF
TITLE ISO-Image erstellen...

:-----
::Purpose: Batch does create an ISO file
::
::CALL: Batch will run separatly and is called by user
::
::Version: 1.0.1
::
::History:
::- Corrected path to etfsboot.com. So changed the following
:: two lines, from:
:: IF "%VVersion%"=="x86" SET bootL=%Programfiles%\Windows AIK\Tools\PETools\x86\etfsboot.com
:: IF "%VVersion%"=="amd64" SET bootL=%Programfiles%\Windows
AIK\Tools\PETools\amd64\etfsboot.com
:: to:
:: IF "%VVersion%"=="x86" SET bootL=%Programfiles%\Windows
AIK\Tools\PETools\x86\boot\etfsboot.com
:: IF "%VVersion%"=="amd64" SET bootL=%Programfiles%\Windows
AIK\Tools\PETools\amd64\boot\etfsboot.com
::
:-----

::Check environment
::Fist of all is checked, if all used environment variables are declared and valid and if
::all used directories do exist. This done by check_umgebung.cmd.
::
::check_umgebung.cmd will assign a value "true" to EnvErr variable, if there is an error in
::working environment and it will assign "false" to EnvErr variable, if everything is alright
::in working evrionment.
::
::If there is an error in working environment (EnvErr=true) au_fehler.cmd is called.
::au_fehler.cmd will display an error message and quit the batch.

CALL check_umgebung.cmd
IF %EnvErr%==true CALL au_fehler.cmd

::DECLARING Variables
::You need to adpat the following SET lines, if your system was not prepared or not prepared
::correctly using preplab.cmd. Remove everything in front of SET and adjust the path behind
:::=". Expect PETools - this is path where oscdimg.exe is located - all pahtes are arbitrary.
::
::---- Start Adjustment-----
::SET PETools="%Programfiles%\Windows AIK\Tools\PETools"
::SET VDVD=D:\VistaWork\DVD
::SET VISO=D:\VistaWork\ISO
::SET VVersion=x86
::----End Adjustment-----

:-----
::
::      MAIN PROCEDURE
::
:-----

::The following tow lines will check, if a 32 bit or 64 bit DVD is used.
::This information is stored in VVersion variable wich was defined by preplab.cmd.
::
::The check is necessary to use the right boot loader file for the right Vista Version
::
::The boot loader files making the created ISO file bootable.
::
IF "%VVersion%"=="x86" SET bootL=%Programfiles%\Windows AIK\Tools\PETools\x86\boot\etfsboot.com
IF "%VVersion%"=="amd64" SET bootL=%Programfiles%\Windows AIK\Tools\PETools\amd64\boot\etfsboot.com

::Now ISO will be created using oscdimg.exe:
::
::-b points to the location of the boot sector file (%bootL%).
::This file will make the DVD bootable. Do not use a space between b and path!
```

```
::
::%bootL% was defined above and will be "x86" or "amd64" depending on if a 32 bit or 64 bit
::DVD will be created.
::
::%VDVD% points to the path, where the Vista DVD files are located. These are the files
::contained by the ISO file later.
::
::%VISO%\Vista%VVersion%.iso points to the path, where the ISO file will be created in and
::defines the name of the ISO, too.
::
::-n makes it possible to use extended file names.
::-m allows to create iso files, larger than CD format.

START "Creating ISO. . ." /WAIT "%PETools%\oscdimg -n -m -b"%bootL%" %VDVD%
%VISO%\Vista%VVersion%.iso
ECHO.
ECHO The ISO file was created. Please press any key to quit. . .
PAUSE > NUL
EXIT
```

au_fehler.cmd

Purpose:

- The au_fehler.cmd is a sub batch displaying an error message, if there is an error in working environment.

```
@ECHO OFF
TITLE Error working environment!

:-----
::Purpose: Batch displays error message and closes calling batch
::
::CALL: Called from other batchs
::
::Version: 1.0
:-----

::This batch is called from other batchs always, if an error is detected inside
::working environment.

::The batch does display an error message, if environment variables and/or
::directories are not setted correctly or do not exist. The calling batch will be
::canceled.
::
::Beside the log file (EnvCheck.log) is opened using notepad.exe, to verify what is
::causing the error.

::
::-----
::      MAIN PROCEDURE
::-----

ECHO.
ECHO There is an error inside your working environment, so batch can't be
ECHO continued and needs to cacle!
ECHO.
ECHO Please have a look into EnvCheck.log ein, to get some more detailed infromation.
ECHO.
ECHO Please press any key to quit . . .
%systemroot%\notepad.exe "EnvCheck.log"
PAUSE >NUL
EXIT
```

build_menu.cmd

Purpose: - build_menu.cmd is a sub batch to build a selection menu generated by informations of a .wim file.

```
@ECHO OFF
TITLE Menu Builder

:-----
::Purpose: Building a selection menu of an .wim file index
::
::Call: Called from other batchs only
::
::Version: 1.0
:-----

::Declaring variables
::You need to adapt the following SET lines, if your system was not prepared or not prepared
::correctly using preplab.cmd. Remove everything in front of SET and adjust the path behind
::"=". Expect ImgX - this is path where imagex.exe is located - all paths are arbitrary.
:----- Start Adjustment-----
::SET ImgX="%Programfiles%\Windows AIK\Tools\x86"
::SET VMount=J:\VistaWork\Mount
::SET VLog=J:\VistaWork\Logs
::SET VDVD=J:\VistaWork\DVD
:-----End Adjustment-----

::
:: _____
::      MAIN PROCEDURE
:: _____

::First of all the number of menu entries has to be acquired. This number is equivalent to
::number of images inside .wim file.
::
::The number of images is stored in "Image Count" property and can be displayed using
::imagex /info.

:Anfang
CLS

::The following FOR loop separates all matches got from FINDSTR looking for "Image Count:"
::inside index.txt in a part (Token) in front of ":" (delims=delimiter=separator string)
::and one part (Token) behind ":". The second Token will be stored inside ICount variable:
::
:: Tok 1      Tok 2
::   |        |
::Image Count: 7
::           |
::           delims
::
::
::Cause there is a space in front of "7" (" 7"), ICount is shorted to the second character
::by SET ICount="ICount:~1% ("7"). Counting starts with 0 in this case. So ICount is even
::"7".

FOR /F "Tokens=2 delims=:" %i IN ('FINDSTR /c:"Image Count:" %VLog%\index.txt') DO SET ICount=%i
SET ICount=%ICount:~1%
ECHO The .wim file does contain %ICount% image(s). Which one of the following
ECHO images you want to mount for editing?
ECHO.

::With the number of images only, user has not enough informations to choose an image.
::The user will need a description or a name in addition to do that.
::
::So the <DESCRIPTION> tag - which is dumped from imagex /info for every image inside wim file -
::will be exploited.
::
::Again a FOR loop is used together with FINDSTR to separate the Tokens. This time different
::delimiters will be used (delims), to cut everything that's not needed.
::
::We need the third Token only, which will be redirected to Menu.txt file.

FOR /F "Tokens=3 delims=<>" %i IN ('FINDSTR /n ".DESCRIPTION." %VLog%\index.txt') DO ECHO %i >>
%VLog%\Menu.txt
```

```

::
::
::Tok 1      Tok 2      Tok 3      Tok 4
::  |        |        |        |
::"  <DESCRIPTION>Windows Vista Ultimate</DESCRIPTION>"
::  |        |        |        |
::  delim    delim    delim    delim
::
::

::Next FINDSTR will display all (".") rows from inside Menu.txt. /n option will precede each
::line an order number.

FINDSTR /n "." %VLog%\Menu.txt

DEL /q %VLog%\Menu.txt

::The menu was builded depending on informations contained inside .wim file. The build_menu.cmd
::will be quited.

GOTO :EOF

```

check_umgebung.cmd

Purpose: - check_umgebung.cmd is a sub batch to check for errors in working environment.

```
@ECHO OFF
TITLE Checking working environment. . .

::-----
::Purpose: Batch does check for errors inside working environment
::
::CALL: Batch will be called from other batchs only.
::
::Version: 1.0
::-----

::This batch is called from other batches to check, if all necessary environment
::variables are set at all and if their values are valid.
::
::Furthermore it checks, if the directories - declared in variables - are existing
::in fact.
::
::It could cause malfunction, if an environment variable would not be set or a
::directory would not exist. It could cause data loss in worst case!

::The batch sets the variable EnvErr=true, if an environment variable is NOT
::valid or a directory does NOT exist.
::
::It will set EnvErr=false, if working environment is alright.
::
::The EnvErr variable can be exploited by the batch, which has called
::check_umgebung.cmd.
::
::The whole check up procedure is logged inside EnvCheck.log file. EnvCheck.log will be
::created in the same directory, where check_umgebung.cmd is located.
::
::To create EnvCheck.log all ECHO commands are redirected into EnvCheck.log.
::This is done by using ">" character. One ">" means, the ECHO command will overwrite
::old entries inside the file. ">>" means the output is added to existing file.
::
::So all ECHO commands inside this batch are for logging only!

::-----
::  MAIN PROCEDURE
::-----

SET EnvErr=false
ECHO Working Environment Check:  > EnvCheck.log

:CheckUmgebung
ECHO. >> EnvCheck.log
ECHO Environment variables are checked... >> EnvCheck.log
ECHO. >> EnvCheck.log

::The variable Count is a counter, which will be increased, if an error is detected.
::IF Count stays on "0", there is no error.
::
::SET >set.txt is writing the list of environment variable, which are shown by SET command
::into set.txt file.
::
::The FOR loop does forward the names of variables which need to be checked to the
::subroutine :CheckU.

SET Count=0
SET >set.txt
FOR %%i IN (VWork VDVD VDistri VLog VMount Vwim VUpdates VSandb VVersion PMgrDir PETools ImgX) DO (
    CALL :CheckU %%i
)

::After checking, if all variables do exist, the set.txt is deleted.
::%Count% will be checked, if it's still on "0". If NOT the subroutine FehlerUmgebung
::will be executed.

DEL /q set.txt
IF NOT "%Count%"=="0" GOTO FehlerUmgebung
ECHO. >> EnvCheck.log
IF "%Count%"=="0" ECHO All necessary environment variables are setted. >> EnvCheck.log

::If %Count% is still "0", batch will be continued here.
```

```

::
::Next the directories declared in variables are checked. Using a FOR loop again they
::are forwarded to CheckV subroutine.
::
::Again %Count% will be increased, if an error is detected.

:CheckVerzeichnis
ECHO. >> EnvCheck.log
ECHO. >> EnvCheck.log
ECHO Checking existnece of directories... >> EnvCheck.log
ECHO. >> EnvCheck.log
SET Count=0

FOR %%i IN (%VWork% %VDVD% %VDistri% %VLog% %VMount% %Vwim% %VUpdates% %VSandb%) DO (
    CALL :CheckV %%i
)

::The directories PMgrDir, PETools und ImgX have to be checked separatly, cause they have
::a space inside path (C:\Programme\Windows AIK\Tools...). The whole path has to be set
::in quotes. This makes it impossible to check NUL.
::IF NOT EXIST "%PMgrDir%\NUL" will not be exploited, just a little like
::IF NOT EXIST "%PMgrDir%\NUL.
::
::To workaroud we will check, if a special file does exist inside these directories.

:CheckWAIKVerz
IF NOT EXIST "%PMgrDir%\pkgmgr.exe" ECHO The directory "%PMgrDir%" does NOT exist! >> EnvCheck.log
IF NOT EXIST "%PMgrDir%\pkgmgr.exe" SET /a Count+=1
IF EXIST "%PMgrDir%\pkgmgr.exe" ECHO The directory "%PMgrDir%" does exist. >> EnvCheck.log

IF NOT EXIST "%PETools%\oscdimg.exe" ECHO The directory "%PETools%" does NOT exist! >> EnvCheck.log
IF NOT EXIST "%PETools%\oscdimg.exe" SET /a Count+=1
IF EXIST "%PETools%\oscdimg.exe" ECHO The directory "%PETools%" does exist. >> EnvCheck.log

IF NOT EXIST "%ImgX%\imagex.exe" ECHO The directory "%ImgX%" does NOT exist! >> EnvCheck.log
IF NOT EXIST "%ImgX%\imagex.exe" SET /a Count+=1
IF EXIST "%ImgX%\imagex.exe" ECHO The directory "%ImgX%" does exist. >> EnvCheck.log

::IF %Count% is NOT "0" batch will execute FehlerVerzeichnis subroutine.

IF NOT "%Count%"=="0" GOTO FehlerVerzeichnis
ECHO. >> EnvCheck.log
IF "%Count%"=="0" ECHO All necessary directories do exist. >> EnvCheck.log

::If there are no errors at all - %Count% is still "0" - only, this last section
::will be executed.
::
::EnvErr is setted to false and batch will be quitted.
::
::The batch calling check_umgebung.cmd will be continued.

ECHO. >> EnvCheck.log
ECHO The working environment was checked successfully! >> EnvCheck.log
SET EnvErr=false
GOTO :EOF

::
:: SUB PROCEDURES & PARTS EXECUTED OPTIONAL
::

::If an error is detected with environment variables, the FehlerUmgebung section is
::executed.
::
::The EnvErr variable is set on true. An error message is displaid and is
::insert into EnvCheck.log log file.
::
::GOTO :EOF will lead back to the calling batch.

:FehlerUmgebung
SET EnvErr=true
ECHO. >> EnvCheck.log
ECHO %Count% environment variables are not setted. So the >> EnvCheck.log
ECHO procedure can't be continued. >> EnvCheck.log
ECHO. >> EnvCheck.log
ECHO Check Control Panel \ System \ Avanced \ >> EnvCheck.log

```

```

ECHO    to find out, if all environment variables were setted correctly. >> EnvCheck.log
ECHO. >> EnvCheck.log
ECHO    You will find more informations about that inside Vista guide >> EnvCheck.log
ECHO    in "Preparation\ Prepare directories". >> EnvCheck.log
ECHO. >> EnvCheck.log
GOTO :EOF

::If one or more of the directories do not exist, the FehlerVerzeichnis section is
::executed.
::
::ausgeföhrt. Die Variable EnvErr wird auf true gesetzt. Eine Fehlermeldung wird ausserdem
::
::The EnvErr variable is set on true. An error message is displaied and is
::insert into EnvCheck.log log file.
::
::GOTO :EOF will lead back to the calling batch.

:FehlerVerzeichnis
SET EnvErr=true
ECHO. >> EnvCheck.log
ECHO    %Count% directories do NOT exist. So the procedure >> EnvCheck.log
ECHO    can'b be continued. >> EnvCheck.log
ECHO. >> EnvCheck.log
ECHO    Please check, if all necessary directories are created >> EnvCheck.log
ECHO    properly on your system! >> EnvCheck.log
ECHO. >> EnvCheck.log
ECHO    You will find more informations about that inside Vista guide >> EnvCheck.log
ECHO    in "Preparation\ Prepare directories". >> EnvCheck.log
ECHO. >> EnvCheck.log
GOTO :EOF

::The subroutine CheckU is called from FOR loop inside CheckUmgebung section.
::
::%1 contains the value of the varialbe wich is assigned by the call
::Call :CheckU %i. During first run of loop it's VWork for example.
::
::FINDSTR is looking inside set.txt - which contains a list of all environment variables -
::for assigned variables and logs it into EnvCheck.log log file.
::
::If FINDSTR does not find the name of assigned variable inside the list, the system
::variable %ERRORLEVEL% is set to 1.
::
::So if %ERRORLEVEL%=1, the assigned variable will be logged as NOT DECLARED into log file
::EnvCheck.log.
::
::Furthermore the variable %Count% will be increased 1, to indicate that the variable is NOT
::setted (declared). So there is an error in working envrionment.

:CheckU
FINDSTR "%1" set.txt >> EnvCheck.log
IF "%ERRORLEVEL%"=="1" ECHO %1= NOT DECLARED! >> EnvCheck.log
IF "%ERRORLEVEL%"=="1" SET /a Count+=1
GOTO :EOF

::The Subroutine CheckV is called from FOR loop in CheckVerzeichnis section.
::%1 includes the value which was assigned by the call Call :CheckV %i.
::
::In the first run it includes the value of environment variable
::%VWork%. So it could be D:\VistaWork for example.
::
::So the IF command during first run could look like this
::IF NOT EXIST D:\VistaWork\NUL.
::
::If a directory does exist, than NUL inside the directory does exist, too.
::
::So if NUL does NOT exist, than the directory does NOT exist at all and an error is
::logged into EnvCheck.log.
::
::Furthermore %Count% variable is increased 1, to indicate that the environment variable
::is not setted and so there is an error.

:CheckV
IF NOT EXIST %1\NUL SET /a Count+=1
IF NOT EXIST %1\NUL ECHO The directory "%1" does NOT exist! >> EnvCheck.log
IF EXIST %1\NUL ECHO The directory "%1" does exist. >> EnvCheck.log
GOTO :EOF

```

BEGINNERS

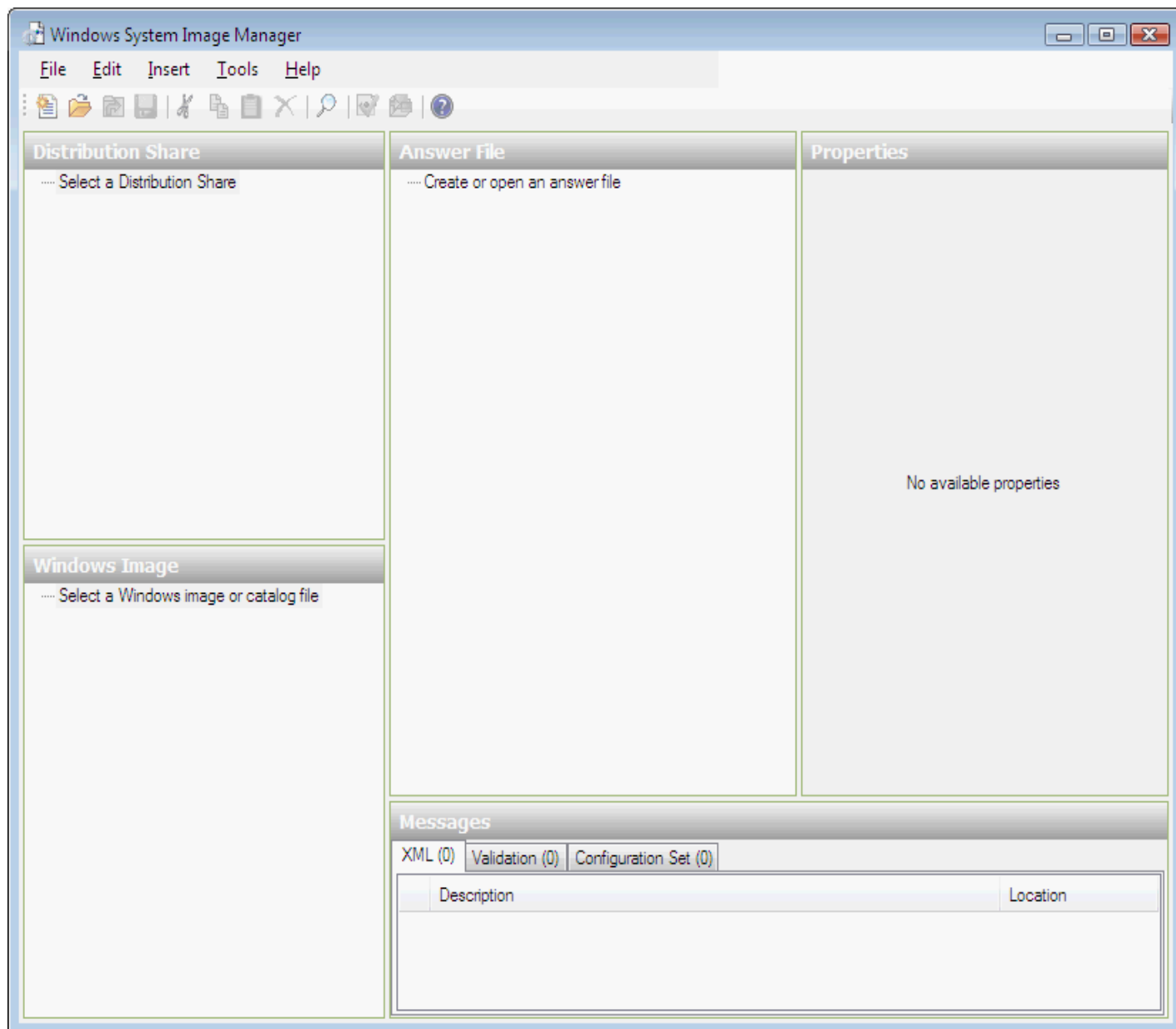
HowTo: Creating an answer file (Autounattend.xml) using WSIM

What we need

1. **Vista DVD** copied to folder on HD
2. **Microsoft's WAIK**

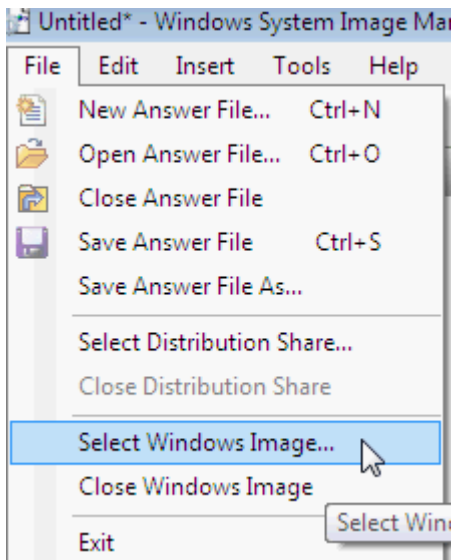
Step 1: Opening WSIM

We will find the **WSIM** under **Start\Program Files\ Microsoft Windows AIK**. After opening the WSIM we will see the following view:

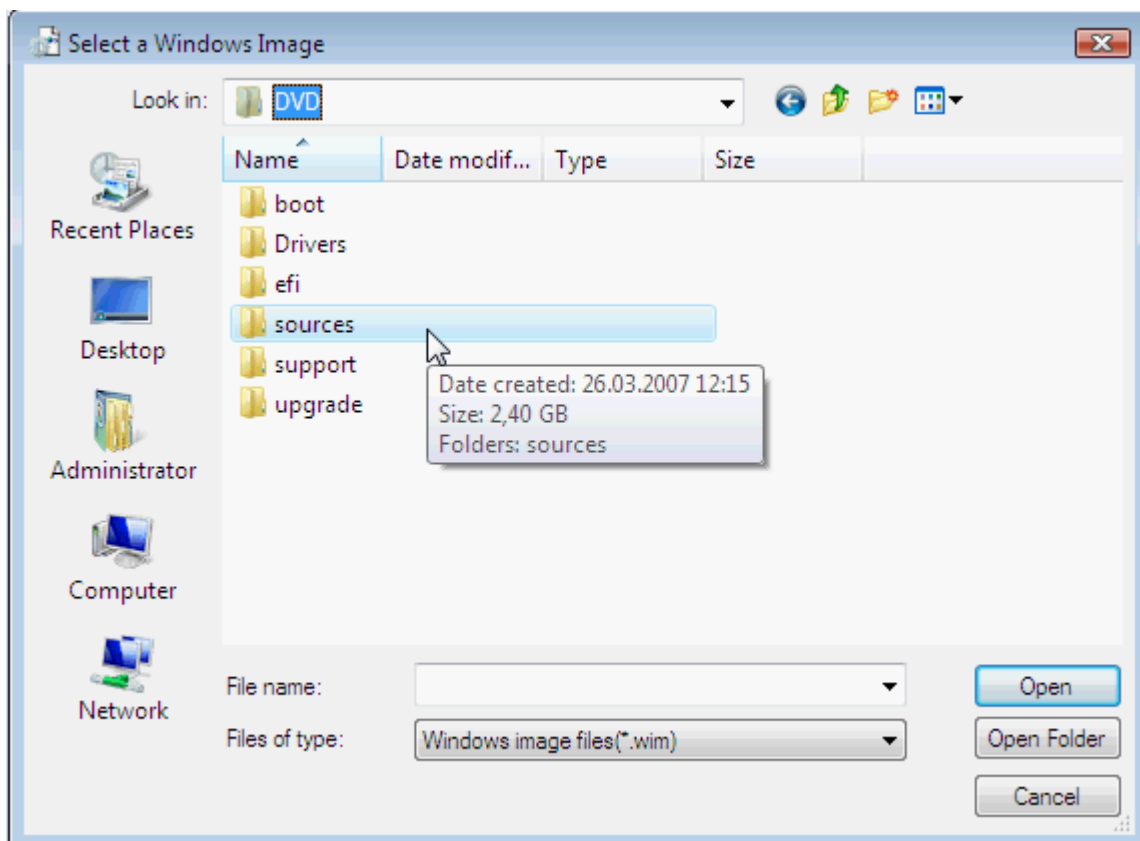


Step 2: Opening Windows image file (install.wim)

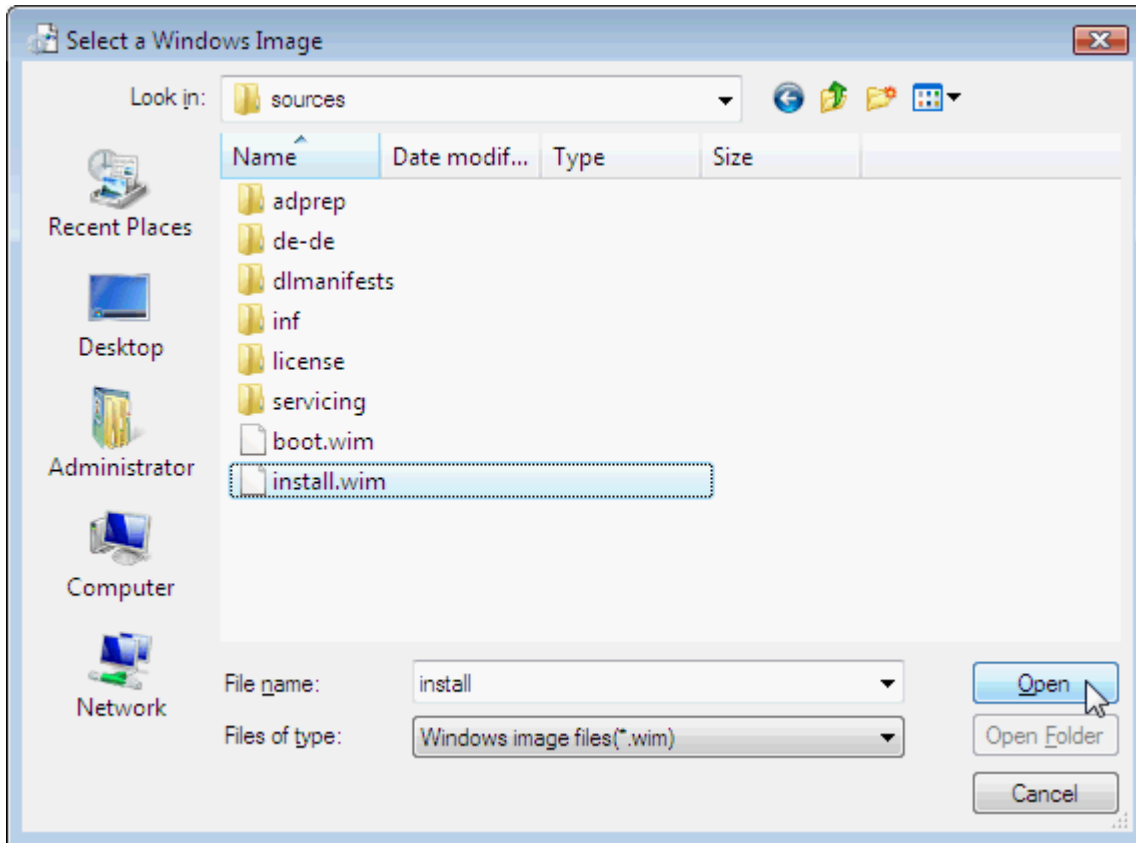
Form **File** menu we choose **Select Windows Image...**



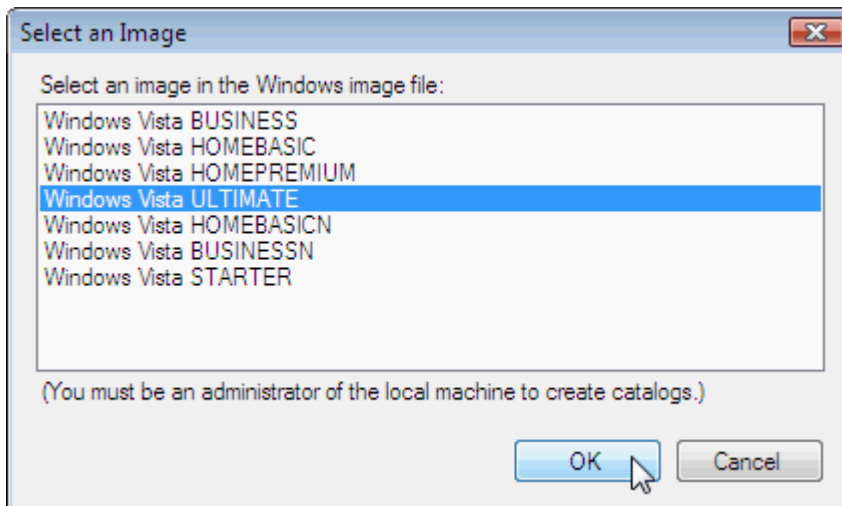
We **browse** to the **directory**, we have copied the **Vista DVD** before to and open **sources** folder:



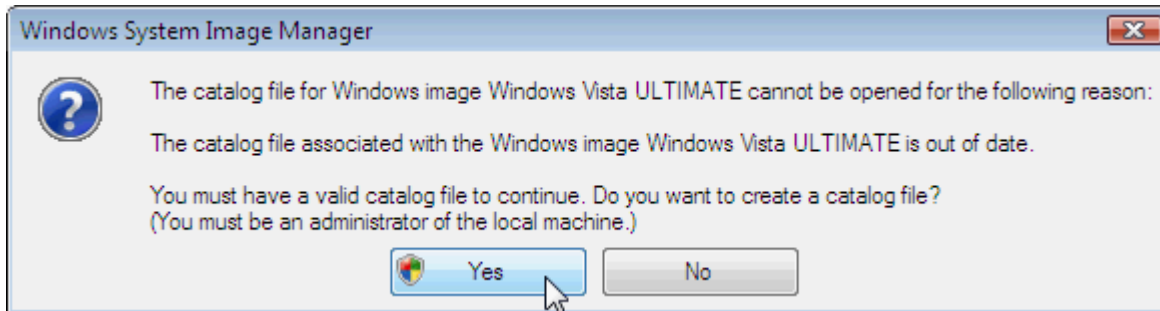
we **left click** the **install.wim** and choose **Open** again.



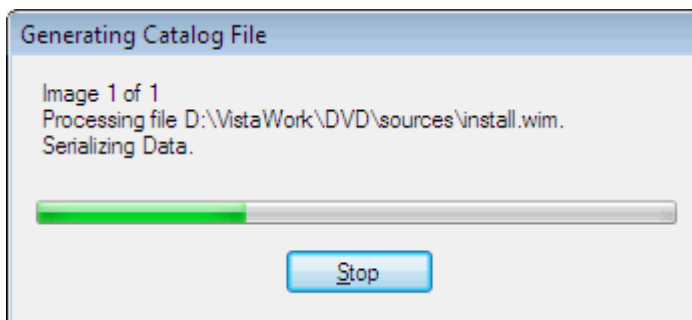
Now we are **prompted to choose** an image. We choose the **version we have the license (Product Key)** for (eg. ULTIMATE) and **click OK**.



The **following question** we **answer** clicking **Yes**.

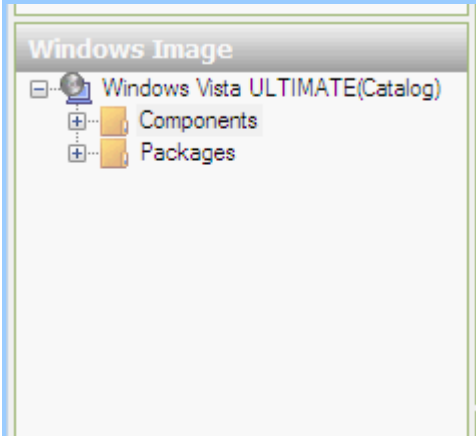


WSIM will start **generating** a **catalog file** of image (install.wim):



NOTE:

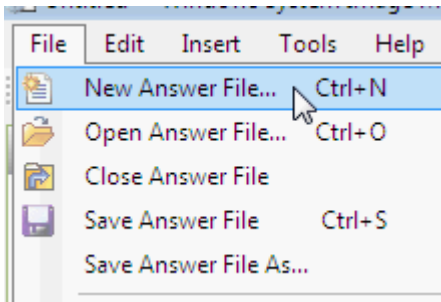
This **procedure will take a while** so **do not cancel it!** Wait till you see the following view in Windows Image pane of WSIM:



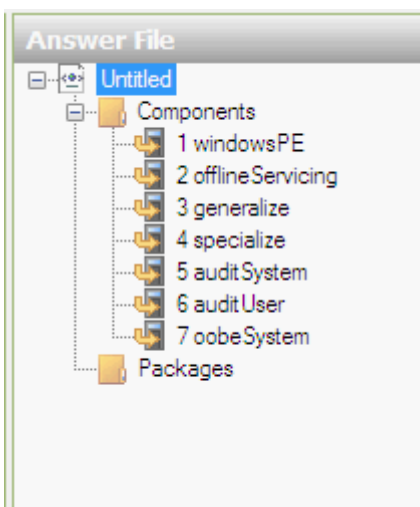
If you have got an error message while generating catalog file, please try the workaround described [here](#).

Step 3: Creating new answer file

From **File menu** we choose **New Answer File...**



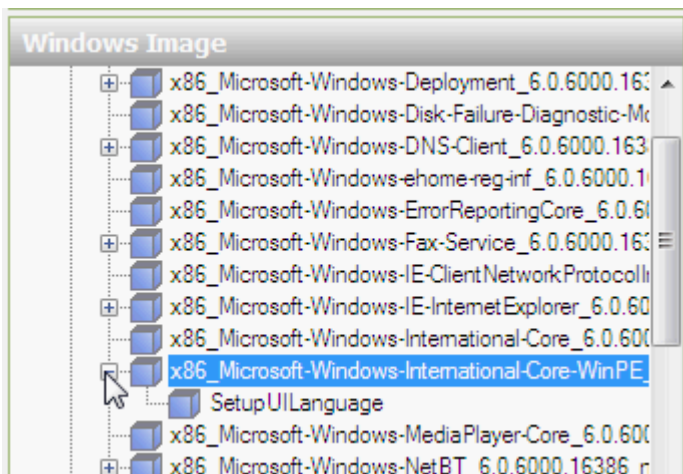
In **Answer File pane** we will see the following entries thereupon:



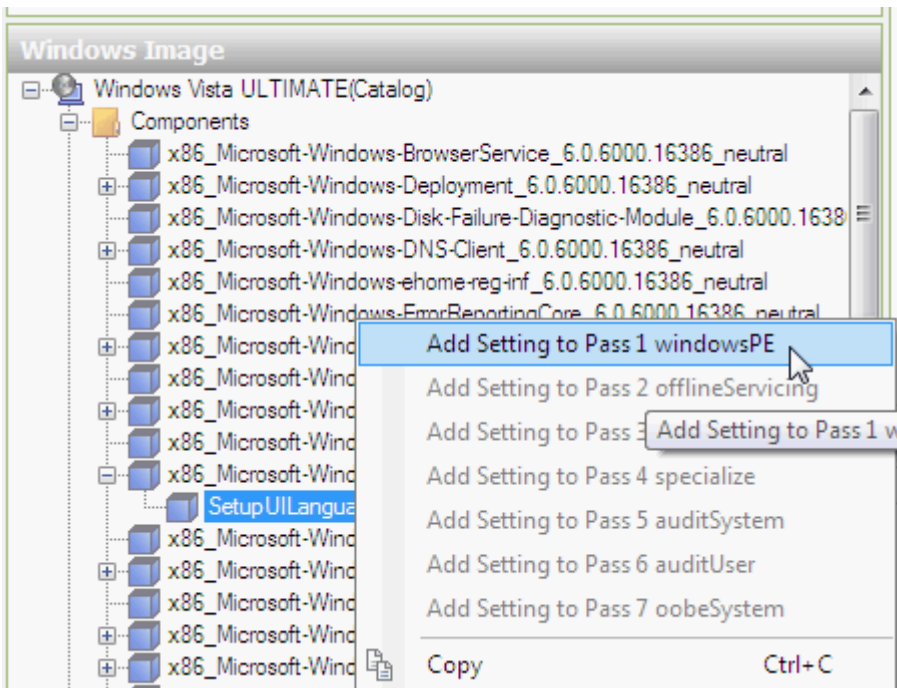
These entries representing the different Vista setup configuration passes.

Step 4: Add components from catalog to answer file

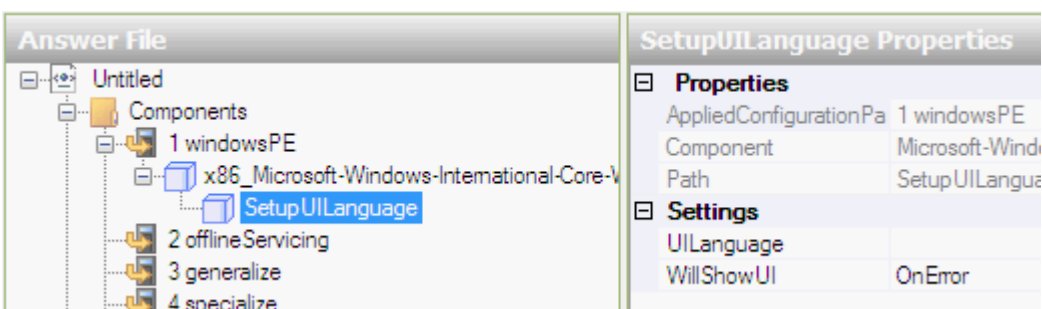
In **Windows Image** pane we **extend** the entry **Components** by clicking on plus in front. From appearing list we choose the component **Microsoft-Windows-International-Core-WinPE** and **extend** it, too:



We **right click** on **SetupUILanguage** and choose **Add Setting to Pass 1 windowsPE**:



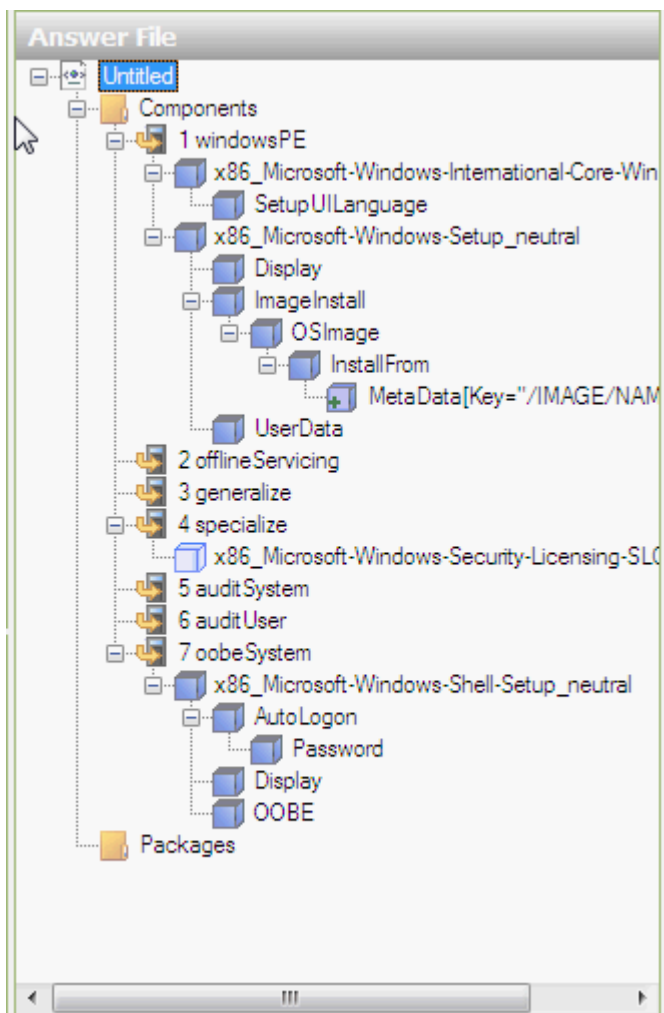
The entry will be transferred to **Answer File** pane, so we can choose it there and change settings.



Before going on editing properties we **add some further components** to our answer file. We ensure, that we **extend** (clicking plus) up to the **listed entry** before we **right click** and **add** it to our **answer file**. So we add the following components:

Component	Pass
Microsoft-Windows-Setup\UserData	-- 1 windowsPE
Microsoft-Windows-Setup\Display	-- 1 windowsPE
Microsoft-Windows-Setup\ImageInstall\OSImage\InstallFrom\MetaData	-- 1 windowsPE
Microsoft-Windows-Security-Licensing-SLC-UX	-- 4 specialize
Microsoft-Windows-Shell-Setup\AutoLogon	-- 7
Microsoft-Windows-Shell-Setup\Display	-- 7
Microsoft-Windows-Shell-Setup\OOBE	-- 7

If we did everything the right way our answer file pane will look like this:



Step 5: Editing component settings in answer file

If we **click** on one of the **entries** in **Answer File pane**, we can **edit** the different **settings** of this component in **Properties pane**. So step by step we adapt the following settings:

windowsPE \ Microsoft-Windows-International-Core-WinPE

Id	x86_Microsoft-Windows-International-Core-WinPE
Settings	
InputLocale	0409:00000409
LayeredDriver	
SystemLocale	en-US
UILanguage	en-US
UILanguageFallback	
UserLocale	en-US

If you want to set **an other language** for your system please **check out here**:

InputLocale: Use the **Hexadecimal Identifier** from **Default Input Locales** table from [list here on Microsoft](#).

LayerdDriver: Use the value described [here on Microsoft](#).

SystemLocale: Use the entry from **System Locale** column of the table **Language Pack Defaults** [listed here on Microsoft](#).

UILanguage: Use the entry from **UILanguage** column of the table **Language Pack Defaults** [listed here on Microsoft](#).

UserLocale: Use the entry from **UserLocale** column of the table **Language Pack Defaults** [listed here on Microsoft](#).

NOTE:

These **properties** will be **used** during **WindowsPE** setup **pass only**! They are not setting the language for the final desktop!

windowsPE \ Microsoft-Windows-International-Core-WinPE \ SetupUI Language

Path	SetupUILanguage
Settings	
UILanguage	en-US
WillShowUI	OnError

UILanguage: Use the entry from **UILanguage** column of the table **Language Pack Defaults** [listed here on Microsoft](#).

NOTE:

These **properties** will be **used** during **WindowsPE** setup **pass only**! They are not setting the language for the final desktop!

windowsPE \ Microsoft-Windows-Setup \ Display

Settings	
ColorDepth	32
HorizontalResolution	1024
RefreshRate	
VerticalResolution	768

Refresh Rate problem

The **refresh rate** is **not settled consciously**. You can try to set to any supported mode of your video adapter, like **60, 75** etc.. However I was **never able to set** this **property** in answer file **successfully**.

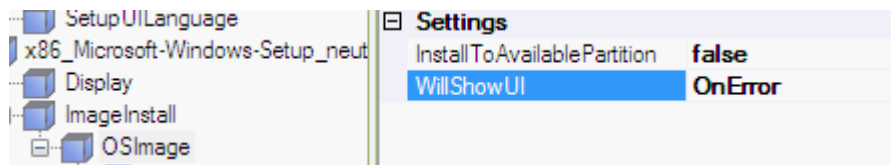
The display was never settled and I've had error messages for display setting in unattended log.

After removing the refresh rate completely, it was setting the right resolution for my video adapter.

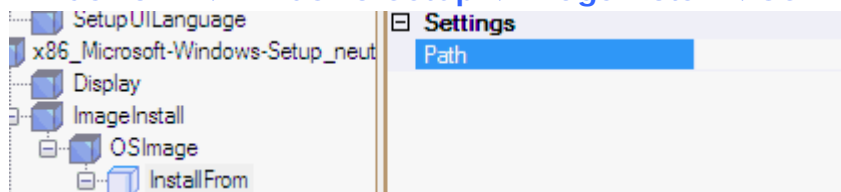
How to set display for end users desktop?

You can set up display settings for every single setup pass, except offlineSevicing and generalize. If you want to adjust the screen settings for end users desktop you need to set them in the last oobe pass that will run on the target system!

windowsPE \ Windows-Setup \ ImageInstall \ OSImage

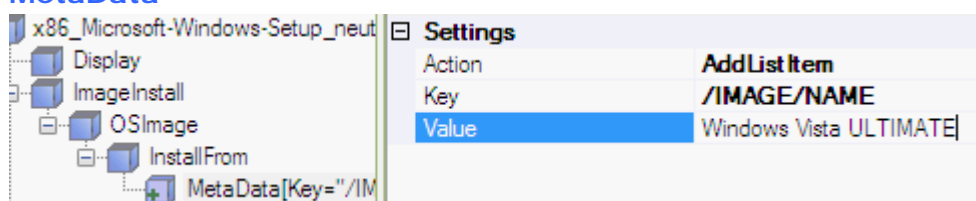


windowsPE \ Windows-Setup \ ImageInstall \ OSImage \ InstallFrom



If your **setup source** is a **DVD** it's **not necessary to fill** something in there. So leave the Path field blank!

windowsPE \ Windows-Setup \ ImageInstall \ OSImage \ InstallFrom \ Metadata



The **Metadata** properties will **point setup to** the right **Vista version inside install.wim**.

What does that mean?


You may have recognized already, that the Vista **DVD does not contain one** of the Vista **versions only**. It **contains all of them**. The different versions are not spanned over different folders. They **are all stored in one** big file - the **install.wim**.

So if we tell setup, that our setup source is install.wim - like we did with InstallFrom Path setting - we **need to point** setup to the **right version** of Vista **inside install.wim**. We have **three options** to point setup to stored version inside install.wim:

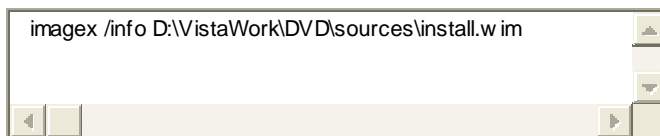
Key	Value
/IMAGE/NAME	<Name>
/IMAGE/INDEX	<Image Index>
/IMAGE/DESCRIPTION	<Description>

The **italic information** we need to fill out with the **right values** now.

Where we can get the Value from?

The **install.wim** does **contain an index** which describes the contents. To show this index we can use **imagex /info**  command.

We click on **Start\Program Files\Microsoft Windows AIK\ the Windows PE Tools Command Prompt**. We type the following command:



D:\VistaWork\DVD\sources\install.wim you need to **replace** with the **path to your install.wim**!

The following informations will show up. If you use an adapted install.wim it will may look a bit different:

```

C:\Programme\Windows AIK\Tools\PETools>imagex /info J:\Uista\UistaRC1-DE-DUD\sources\install.wim

ImageX Tool for Windows
Copyright (C) Microsoft Corp. 1981-2005. All rights reserved.

WIM Information:
-----
GUID:           {9838c7d9-eece-46a8-8119-fec09d12de8c}
Image Count:    7
Compression:    LZX
Part Number:    1/1
Attributes:     0xc
                 Integrity info
                 Relative path junction

Available Image Choices:
-----
<WIM>
<TOTALBYTES>2374302990</TOTALBYTES>
<IMAGE INDEX="1">
<NAME>Windows Vista BUSINESS</NAME>
<DESCRIPTION>Windows Vista Business</DESCRIPTION>
<IMAGE>BUSINESS</IMAGE>
<WINDOWS>
<ARCH>0</ARCH>
<PRODUCTNAME>Microsoft<< Windows<< Operating System</PRODUCTNAME>
<HAL>acpiapic</HAL>
<PRODUCTTYPE>WinNT</PRODUCTTYPE>
<PRODUCTSUITE>Terminal Server</PRODUCTSUITE>
<LANGUAGES>
<LANGUAGE>de-DE</LANGUAGE>

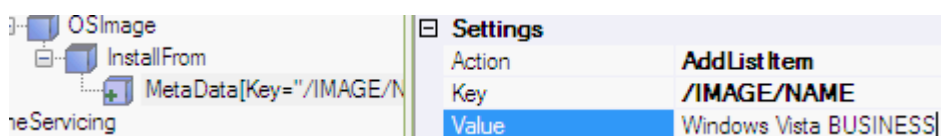
```

This **example** here does **give** us the **following values**:

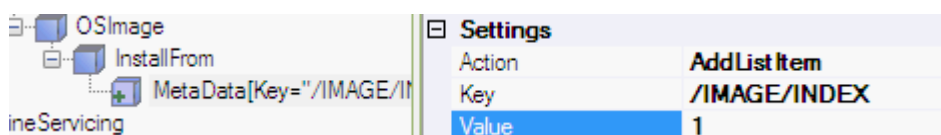
Key	Value
/IMAGE/NAME	Windows Vista BUSINESS
/IMAGE/INDEX	1
/IMAGE/DESCRIPTION	Windows Vista BUSINESS

You could **use ONE** of these three **options** now, to **point** to **Windows Vista BUSINESS** version inside install.wim:

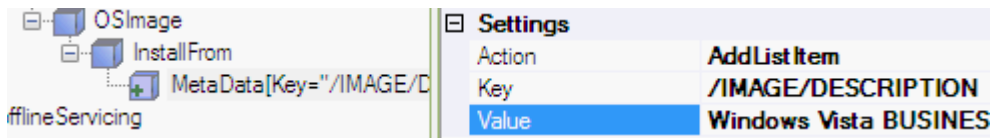
Option 1: Use /IMAGE/NAME



Option 2: Use /IMAGE/INDEX



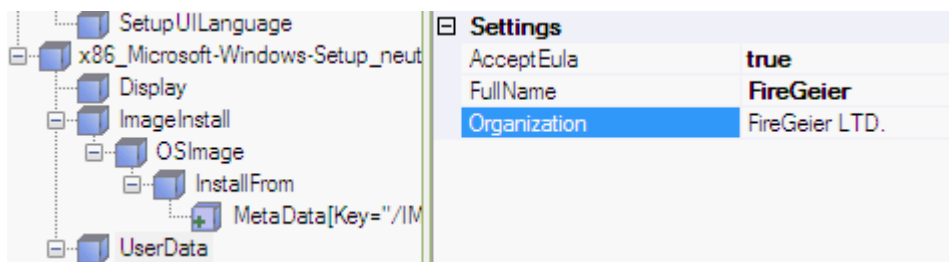
Option 3: Use /IMAGE/DESCRIPTION



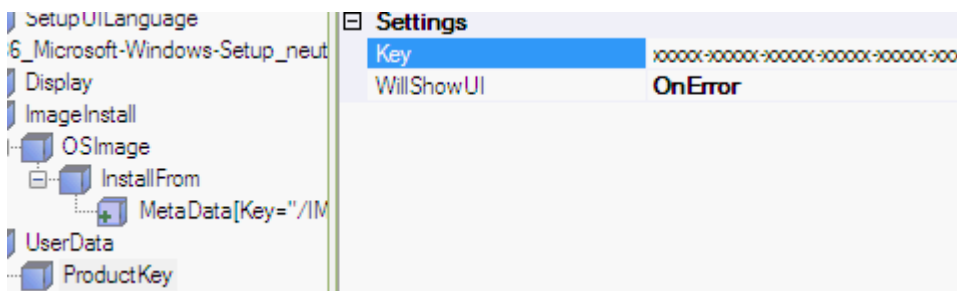
NOTE:

You can **just** use **one** of these **options at the same time**. Ensure to have **one MetaData** section in your Autounattend.xml **only**!

windowsPE \ Windows-Setup \ UserData

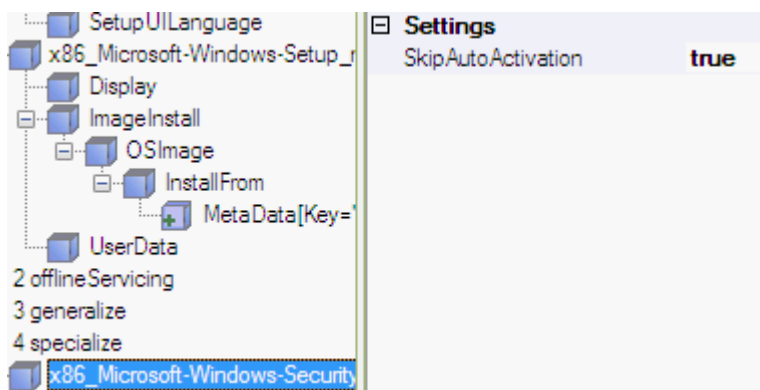


windowsPE \ Windows-Setup \ UserData \ ProductKey



You have to **insert your key** here.

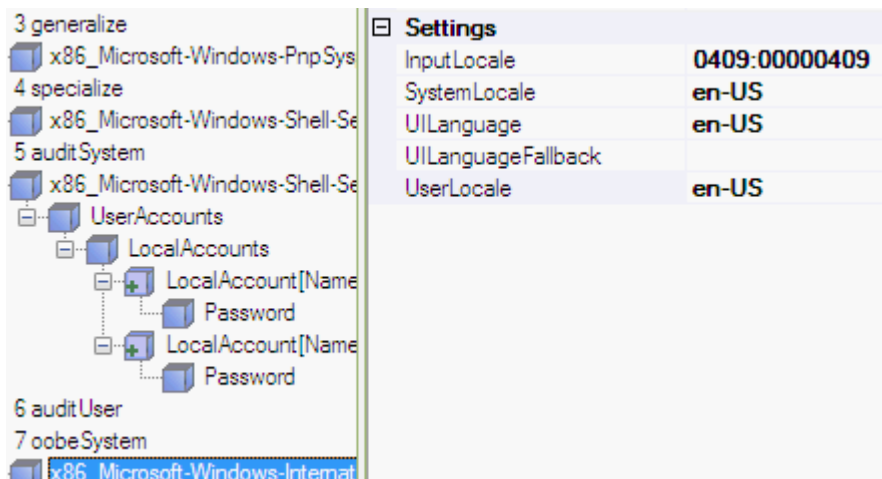
specialize \ Windows-Security-Licensing-SLC-UX



NOTE:

I highly recommend to use this setting while experimenting with your unattended setups. The setting will avoid, that your Vista will be activated automatically, once it has found an internet connection on your system.

oobeSystem \ Windows-International-Core



If you want to set **an other language** for your system please **check out here**:

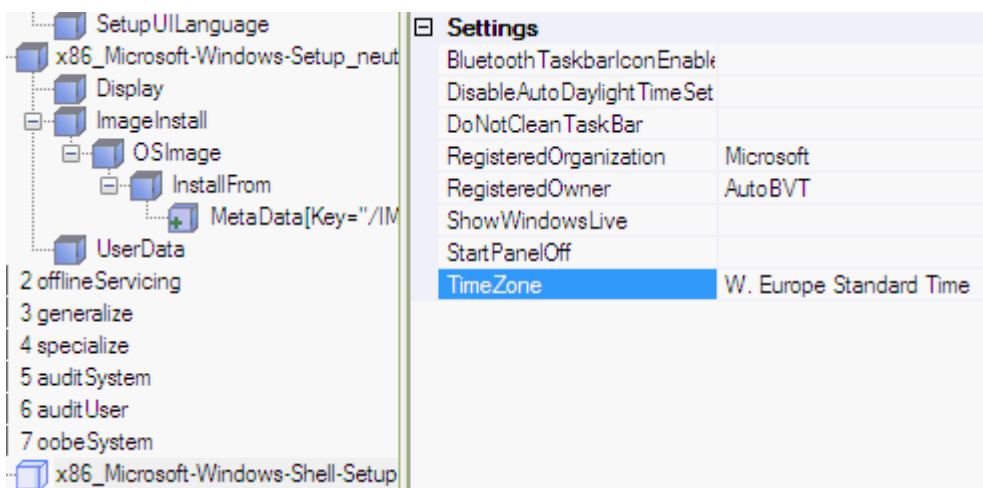
InputLocale: Use the **Hexadecimal Identifier** from **Default Input Locales** table from [list here on Microsoft](#).

SystemLocale: Use the entry from **System Locale** column of the table **Language Pack Defaults** [listed here on Microsoft](#).

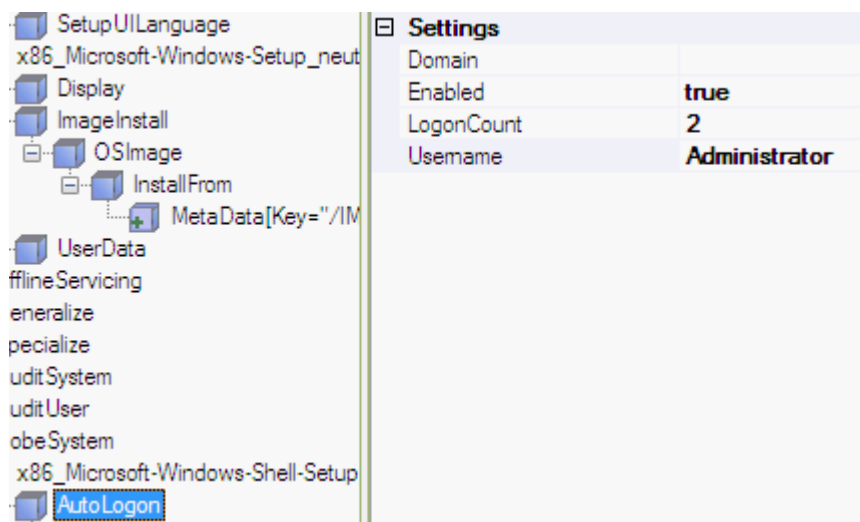
UILanguage: Use the entry from **UILanguage** column of the table **Language Pack Defaults** [listed here on Microsoft](#).

UserLocale: Use the entry from **UserLocale** column of the table **Language Pack Defaults** [listed here on Microsoft](#).

oobeSystem \ Windows-Shell-Setup



oobeSystem \ Windows-Shell-Setup \ AutoLogon



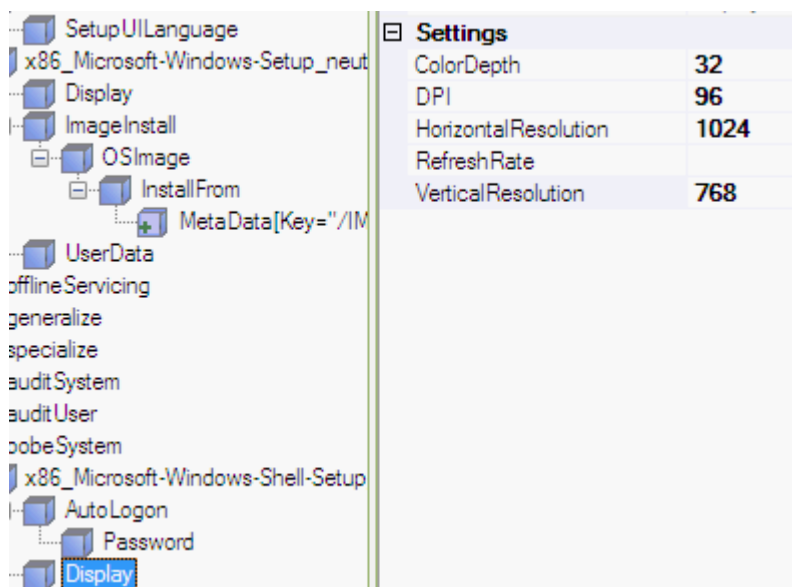
NOTE:

If you set "Administrator" as username, the **build in administrator** account will be **enabled**.

Always use the English word Administrator, whatever language version you're using! **Only the English word Administrator will enable the main administrator account!**

Please don't try to use an other than the Administrator account for this basic Autounattend.xml!

oobeSystem \ Windows-Shell-Setup \ Display



Refresh Rate problem

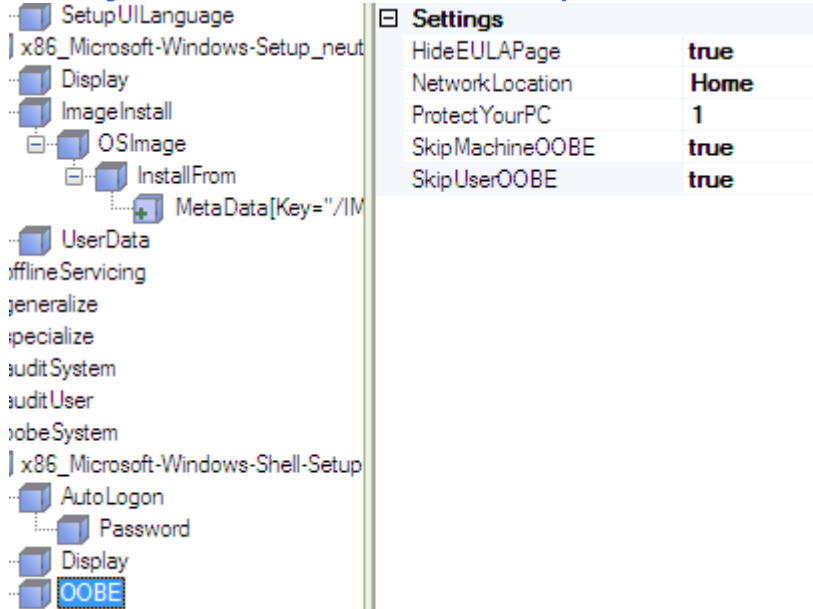
The **refresh rate** is **not settled consciously**. You can try to set to any supported mode of your video adapter, like **60, 75 etc..** However I was **never able to set this property** in answer file **successfully**. The display was never settled and I've had error messages for display setting in unattended log.

After removing the refresh rate completely, it was **setting the right resolution** for my video adapter.

How to set display for end users desktop?

You can set up display settings for every single setup pass, except offlineServicing and generalize. If you want to adjust the screen settings for end users desktop you need to set them in the last oobe pass that will run on the target system!

oobeSystem \ Windows-Shell-Setup \ OOBE

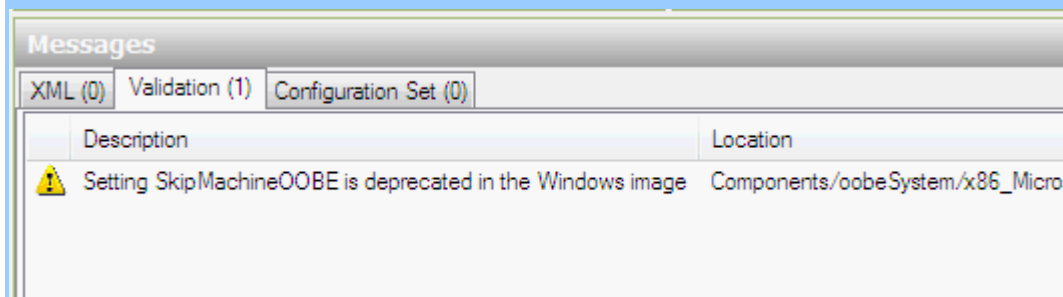


The screenshot shows the Windows Setup configuration tool. On the left, a tree view displays the setup components, with 'oobeSystem \ Windows-Shell-Setup \ OOBE' selected. On the right, the 'Settings' table is displayed.

Setting	Value
HideEULAPage	true
NetworkLocation	Home
ProtectYourPC	1
SkipMachineOOBE	true
SkipUserOOBE	true

NOTE:

If you use <SkipMachineOOBE> settled to **true**, you'll get the **following warning**, once you validate or save your answer file:



This is a **warning only!** You can **ignore it for the moment**. It **will not prevent** this basic **unattended** from **working correctly!**

OOBE background information

What is OOBE?

The **OOBE** (**O**ut **O**f **B**ox **E**xperience) are the first few introducing windows, once you're booting a fresh installed system. You'll be asked for the language and regional settings, for the user account you want to create etc..

What is SkipMachineOOBE property doing?

If you set SkipMachineOOBE property to true, the OOBE windows will be suppressed.

Why SkipMachineOOBE is deprecated?

Microsoft does not recommend to use SkipMachineOOBE cause your system could end up in an unusable state. Especially if you do not set up an Administrator account in your answer file, you will not be able to log on to the system at all once the unattended setup has finished.

How to bypass OOBE properly - how to automate OOBE?

The following document on Microsoft TechNet will describe which properties you have to setup to automate the OOBE screens:

Settings to Use for Automating Windows Welcome

Known issues automating OOBE

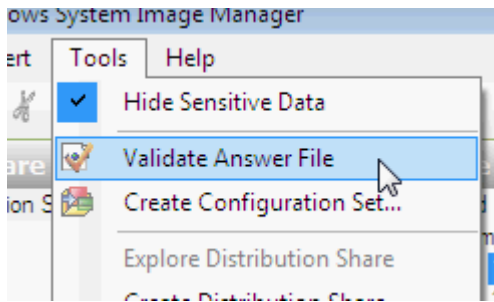
Even if the method described above will **automate** the **OOBE** screen, **there are** some **known issues** using it:

What ever you will set for **NetworkLocation**, it will prompt you in your final installation to choose the network. You need to [set the network location using a regtweak](#).

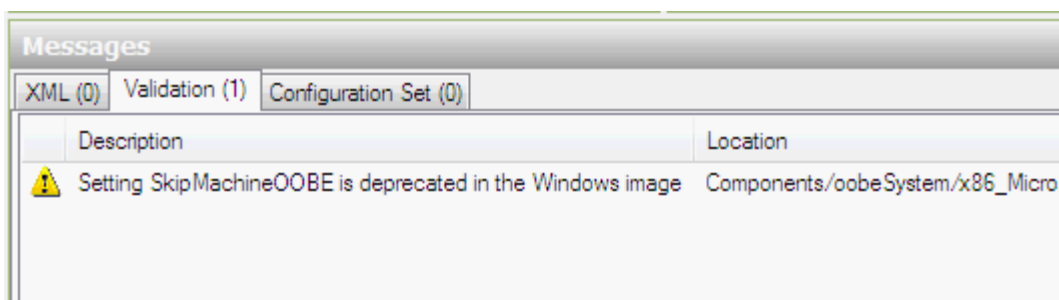
For more informations about **OOBE problems and workarounds** read the [informations on Microsoft TechNet forum](#).

Step 6: Validating answer file

Now we will **validate** the answer file. We click on **Tools menu** and choose **Validate Answer File**.



If **everything went right**, the **following message** will be shown:



If you get an error message (red point with white cross), you have to verify all the settings again.

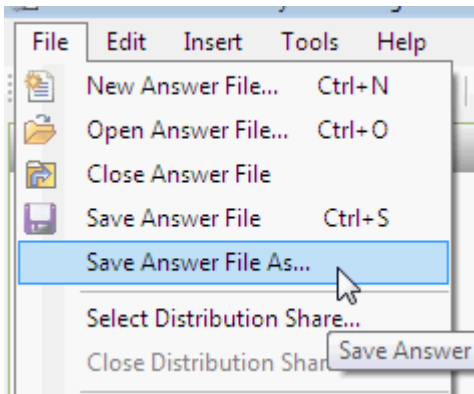
NOTE:

Validating the answer file **successfully does NOT guarantee** that it will run **in fact without** any **problems**. The validation checks for **syntax errors** basically!

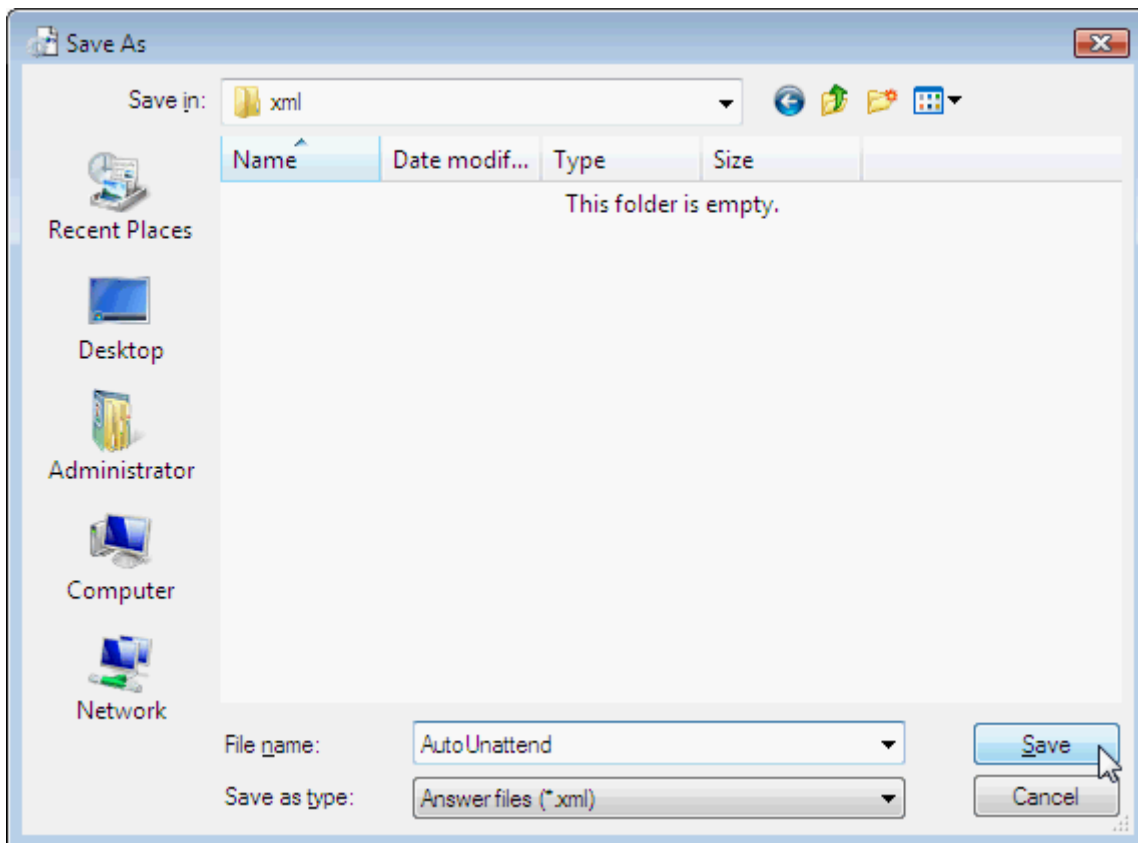
For advanced error checking please read [here](#).

Step 7: Saving answer file as Autounattend.xml

From **File** menu we choose **Save Answer File As...**



We **browse** to **any folder** and insert **Autounattend** as file name. We need to **ensure** just to enter **Autounattend** as name **only** without xml. The extension is already preselected.



Step 8: Copying Autounattend.xml to removable media (Floppy, USB Flash Drive, DVD)

Now we **copy** over the **Autounattend.xml** to **any removable media** like USB flash drive for example. We **put** it to the **root** of the media.

If you want to use **Autounattend.xml** on **Vista DVD** **copy** it to the **root** of the **directory**, where you've **copied** the **Vista DVD** to and **create an ISO** file than.

Step 9: Executing Vista setup

Now we are ready to start our first unattended Vista setup. Just put the removable media and the Vista DVD into your target system and hit a key, when you're prompted.

NOTE:

This is a very simple unattended setup. It includes only basic components and you still have to configure the disk manually.

Known issue: Autounattend.xml seems not to be executed

There were a few threads coming up on MSFN and other forums, discussing where to place Autounattend.xml on DVD. Most people say, it should be placed into the root of DVD, others were using it successfully inside sources folder of DVD.

Vista setup follows a special order, to look for Autounattend.xml described in the document ["Methods for Running Windows Setup"](#) on MS TechNet.

If you have problems getting your Autounattend.xml detected on the root, put it to the sources folder. One or the other alternate supposed to work on your system. I have not found about a rule of this behavior yet. Seems to be, that VMs like having it inside sources.

Partitioning

HowTo: Partitioning

Introduction

Different from all other unattended Windows setups before, Vista unattended setup allows to configure the hard drive automatically. But what sounds great in the first moment can end up in a trial and error nightmare, cause it's not that easy to understand the way of partitioning. So I highly recommend to read the following carefully, even it's a longer text and a bit of theory.

At least it's easy once you got it!

What we need

1. **WAIK** installed
2. **Basic Autounattend.xml** like described [here](#) before

WARNING

ATTENTION! In the following we will create an answer file that will **DELETE ALL EXISTING PARTITIONS** on the hard drive - Disk 0 inside our system. **ALL DATA** on relevant partition **WILL GET LOST!!!**

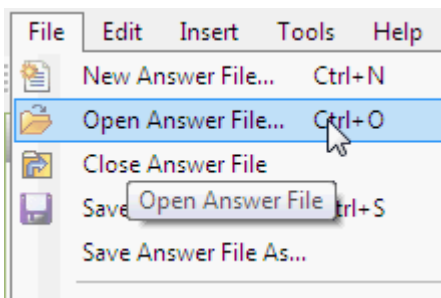
If you have more than one physical hard drive installed inside your computer, I highly recommend to disconnect the ones that you don't use to install Vista on!

Partitioning Rule 1

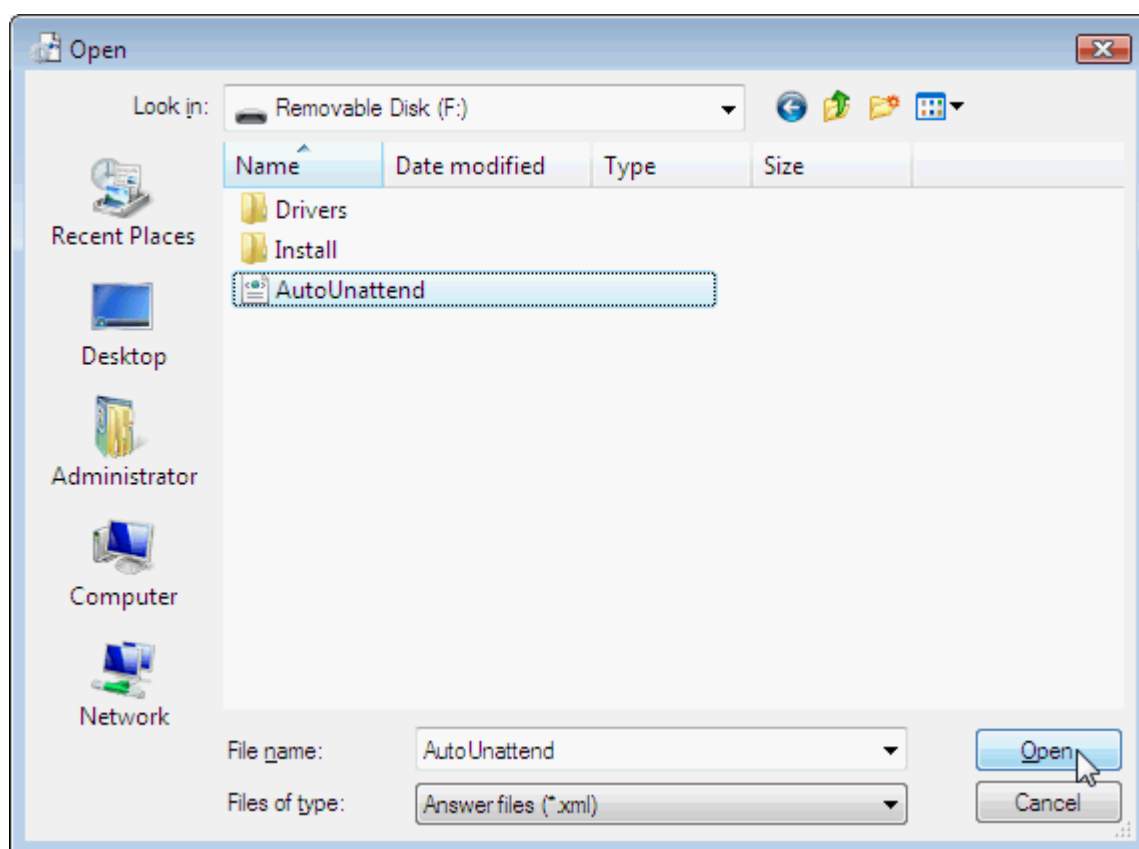
Ensure to make a **backup** of your **data** **before starting** any **automated partitioning** operations.

Step 1: Add DiskConfiguration components to Autounattend.xml

First of all we **open WSIM** and choose **Open** from **File** menu:



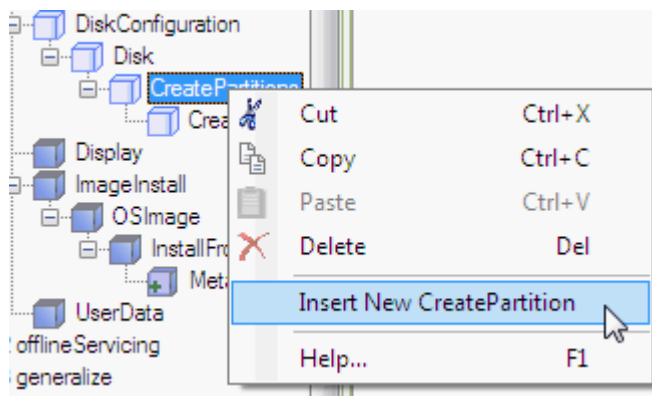
We **browse** to our **Autounattend.xml** and select it to open:



Now we **add** the **following components** to our answer file **by right clicking them** in Windows Image pane:

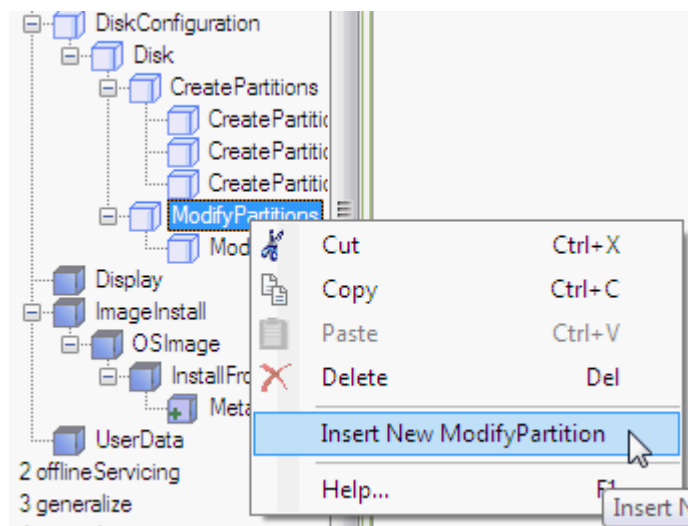
Component	Pass
Microsoft-Windows-Setup\DiskConfiguration\Disk\CreatePartition	--> 1 windowsPE
Microsoft-Windows-Setup\DiskConfiguration\Disk\ModifyPartition	--> 1 windowsPE
Microsoft-Windows-Setup\ImageInstall\OSImage\InstallTo	--> 1 windowsPE

As soon as we have added the component we **right click CreatePartitions** in **Answer File pane** and choose **Insert New CreatePartition**:

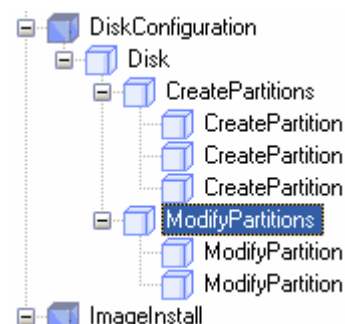


We **repeat this**, so we get **three CreatePartition** entries.

Next we **right click ModifyPartitions**



and choose **Insert New ModifyPartition**. We do it **once** this time **only!** So we end up with **following DiskConfiguration** entries:

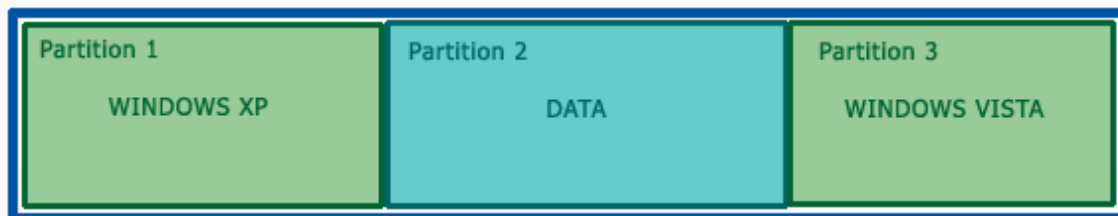


Step 2: Setting Up Disk Configuration Properties

To set up the properties of disk configuration, **a bit of theory** is inalienable **now**. **If you know everything about** partitioning **already** you can skip that section and **go on** with **Setting up CreatePartition and ModifyPartition**.

What is a Partition?

A partition is a **separate independent section on one physical drive**, which will be **treated like a physical drive** by operating systems. It will have an one drive letter and amount of space:



What is the use of Partitions?

Like you can see in the example above, you can create partitions to use **different operating systems** on the same physical drive or/and you can **keep** your **data separated** from your operating system and program files. And partitioning offers the option to use **alternate file systems** - like FAT, NTFS etc. - on the same physical drive.

Partition Types

There are two different types of partitions:

Primary Partitions

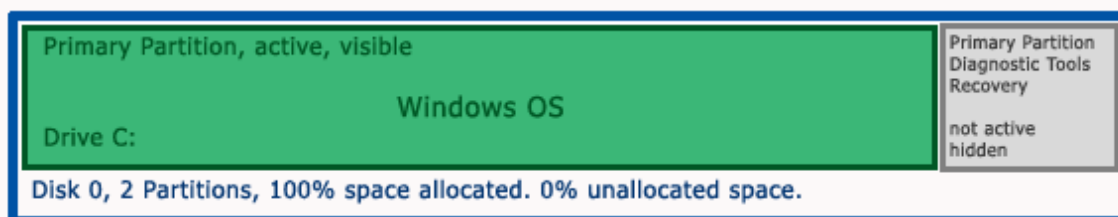
Primary partitions are used, to **contain** and to **start (boot)** an operating system. So a hard drive **must have at least one primary** partition.

It **can have up to four** primary partitions if necessary. **One drive letter** will be attached per primary partition.

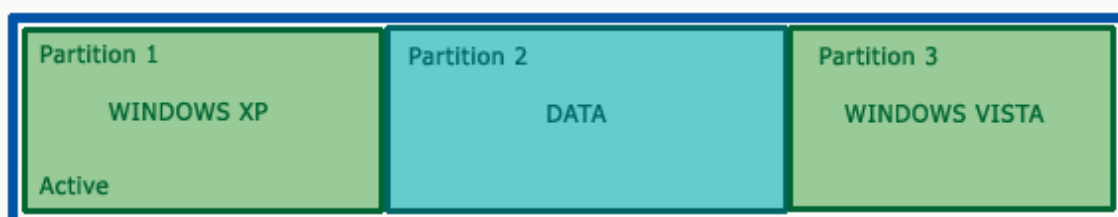
Partitioning Rule 2

A physical hard drive MUST have at least one primary partition.

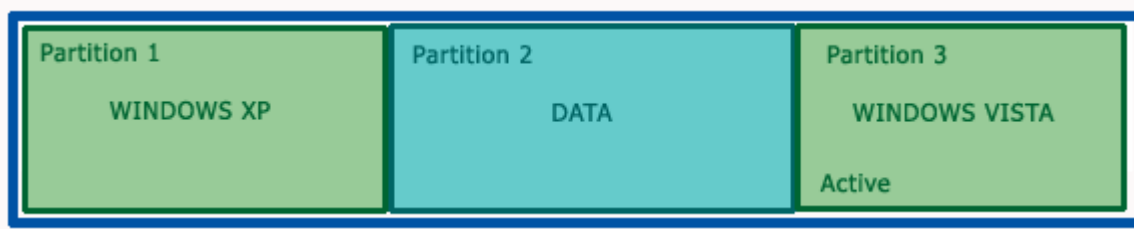
Primary partitions **can hold data** certainly, **too**. So most of manufacture shipped computers have one or two partitions and are looking like this:



Like we see on the scheme above a primary partition can be on **two different states** - **active** and **not active (inactive)**. **Active marks** the primary partition as **bootable**. So this is the one which will be started from, once we power on our computer:



Here **Windows XP (Partition 1)** will be started. On the **next** example it will boot **Windows Vista** on Partition 3:



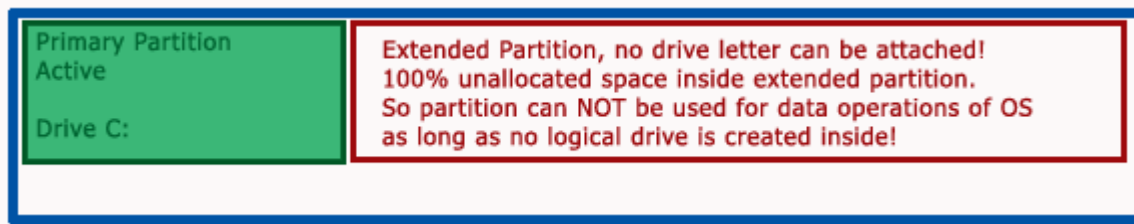
Partitioning Rule 3

One physical drive can always have **one active partition only**.

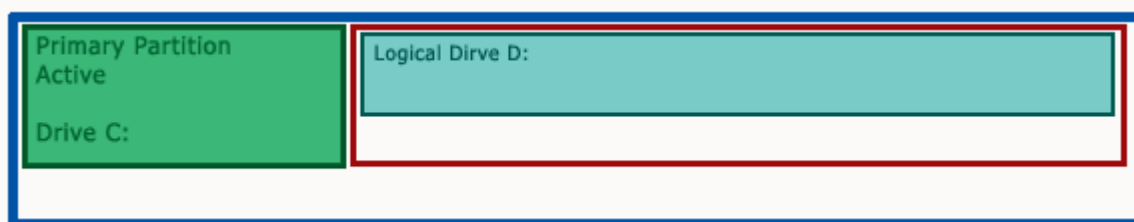
Activating of partitions is **done by** little software programs - almost called **boot manager**. The user will take notice of the boot manager, if he is **prompted to choose, which operating system** he wants to start. Depending on users choice boot manager sets the one or the other partition to **Active**.

Extended Partitions and Logical Drives

Extended partitions are **almost used to store data**. An extended partition is a container only. That means it will allocate an amount of space only but no drive letters will be attached to it. So an operating system can't read or write to it:



To make it readable and writeable an other partitioning object needs to be created - the logical drive(s). We can attach a drive letter to a logical drive and so it becomes useable for the operating system. That means we need three partitioning objects to create two useable partitions - a primary partition, an extended partition and a logical drive inside the extended partition:



An extended partition **can contain more than one logical drive**, if necessary.

Partitioning Rule 4

Extended partitions are **container objects** only. Create **logical drives inside** the extended partition **to let it become useable!**

Setting up CreatePartition and ModifyPartition

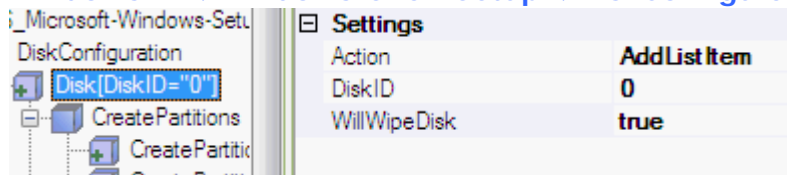
Like we have seen at the beginning our answer file has **two different** disk configuration **components** - **CreatePartition** and **ModifyPartition**.

CreatePartition

CreatePartition is used only to **create new** or **delete existing partitioning objects** (primary, extended and logical). You can only create new partitioning objects, if there is unallocated space on your hard drive.

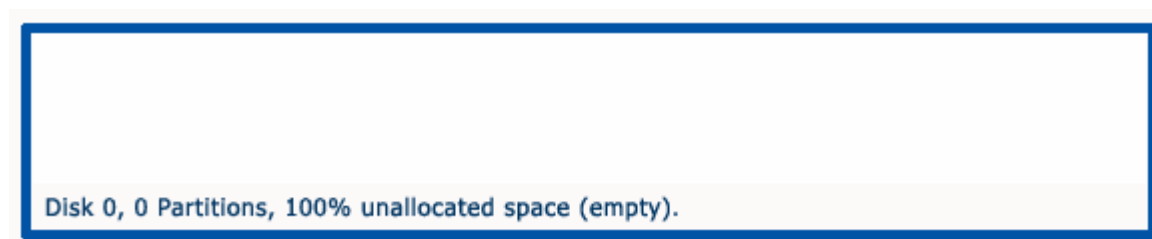
We will insert a command now, to clear the disk, so we know for sure that we have the whole disk space available. We click on **Disk(DiskID="0")** in answer file pane:

windowsPE \ Windows-Shell-Setup \ DiskConfiguration \ Disk("0")



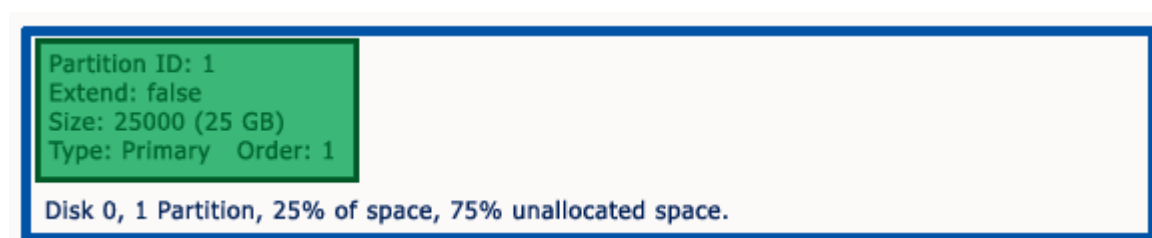
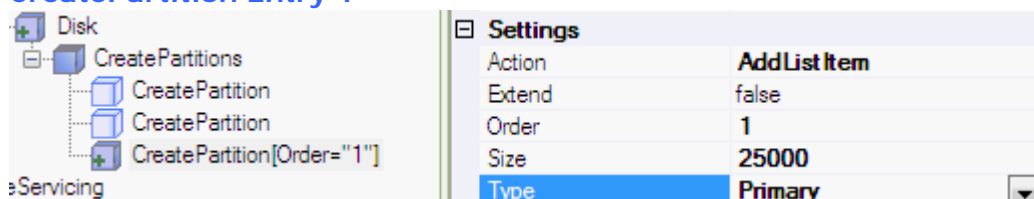
WillWipeDisk on **true** will **DELETE ALL PARTITIONS** on relevant hard drive. So **ALL DATA ON THIS HARD DRIVE WILL GET LOST!**

So now our hard drive will look like this:



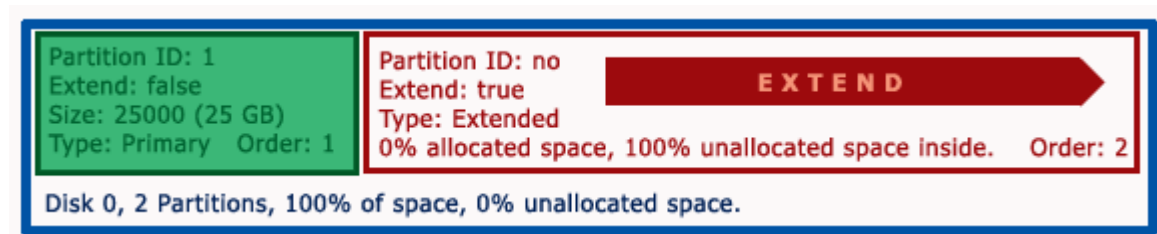
We will **fill out** the three **CreatePartition** settings like the **following** show. The **scheme beneath** is showing the **effect of** the **setting**:

CreatePartition Entry 1



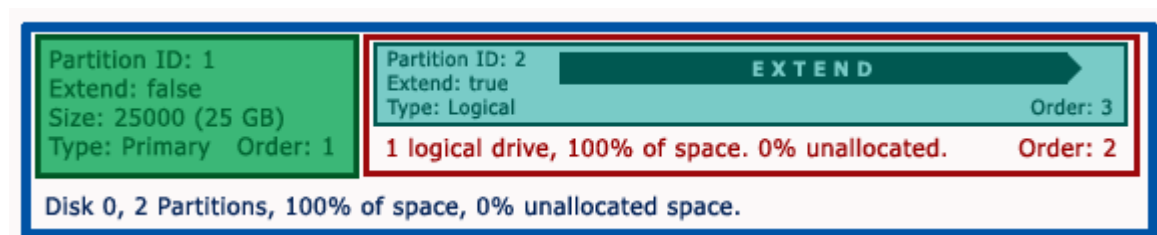
CreatePartition Entry 2

oft-Windows-Setu nfiguration k CreatePartitions	<div>Settings</div> <table><tr><td>Action</td><td>AddListItem</td></tr><tr><td>Extend</td><td>true</td></tr><tr><td>Order</td><td>2</td></tr><tr><td>Size</td><td></td></tr><tr><td>Type</td><td>Extended</td></tr></table>	Action	AddListItem	Extend	true	Order	2	Size		Type	Extended
Action	AddListItem										
Extend	true										
Order	2										
Size											
Type	Extended										



CreatePartition Entry 3

soft-Windows-Setu nfiguration sk CreatePartitions	<div>Settings</div> <table><tr><td>Action</td><td>AddListItem</td></tr><tr><td>Extend</td><td>true</td></tr><tr><td>Order</td><td>3</td></tr><tr><td>Size</td><td></td></tr><tr><td>Type</td><td>Logical</td></tr></table>	Action	AddListItem	Extend	true	Order	3	Size		Type	Logical
Action	AddListItem										
Extend	true										
Order	3										
Size											
Type	Logical										



Size or Extend

It's important to understand the function and **dependency of Size and Extend**. You may have recognized that we have used **Size OR Extend** above. If not please have a look at it again.

Using **Size** we can **setup a fixed size** of the partitioning object. **Extend** will dispose setup to **allocate the rest of space** to the partitioning object. The **logical drive space is limited by the size of the extended partition** thereby certainly.

It's necessary to use **Size OR Extend** for a partitioning object. You can **NOT use both settings** at the **same time**. Otherwise setup will fail!

Partitioning Rule 5

Use **Size OR Extend**. Never use **Size AND Extend** setting for a partitioning object at the same time!

ModifyPartition

ModifyPartition is used to **change settings of an existing partition**. These settings can be the **label**, the **file format**, the **drive letter** etc.. If you have created a partition using CreatePartition you need to use ModifyPartition to setup the label, the file format and the drive letter:

ModifyPartition Entry 1

Microsoft Windows Setup Configuration k CreatePartitions + CreatePartition + CreatePartition + CreatePartition ModifyPartitions + ModifyPartition + ModifyPartition	<table><tr><th colspan="2">Settings</th></tr><tr><td>Action</td><td>AddListItem</td></tr><tr><td>Active</td><td>true</td></tr><tr><td>Extend</td><td>false</td></tr><tr><td>Format</td><td>NTFS</td></tr><tr><td>Label</td><td>Vista_OS</td></tr><tr><td>Letter</td><td>C</td></tr><tr><td>Order</td><td>1</td></tr><tr><td>PartitionID</td><td>1</td></tr></table>	Settings		Action	AddListItem	Active	true	Extend	false	Format	NTFS	Label	Vista_OS	Letter	C	Order	1	PartitionID	1
Settings																			
Action	AddListItem																		
Active	true																		
Extend	false																		
Format	NTFS																		
Label	Vista_OS																		
Letter	C																		
Order	1																		
PartitionID	1																		

Partition ID: 1 Active
Format: NTFS
Label: Vista_OS
Letter: C Order: 1

1 logical drive, 100% of space. 0% unallocated.

Disk 0, 2 Partitions, 100% of space, 0% unallocated space.

ModifyPartition Entry 2

Microsoft Windows Setup Configuration k CreatePartitions + CreatePartition + CreatePartition + CreatePartition ModifyPartitions + ModifyPartition + ModifyPartition	<table><tr><th colspan="2">Settings</th></tr><tr><td>Action</td><td>AddListItem</td></tr><tr><td>Active</td><td>false</td></tr><tr><td>Extend</td><td>false</td></tr><tr><td>Format</td><td>NTFS</td></tr><tr><td>Label</td><td>Data</td></tr><tr><td>Letter</td><td>D</td></tr><tr><td>Order</td><td>2</td></tr><tr><td>PartitionID</td><td>2</td></tr></table>	Settings		Action	AddListItem	Active	false	Extend	false	Format	NTFS	Label	Data	Letter	D	Order	2	PartitionID	2
Settings																			
Action	AddListItem																		
Active	false																		
Extend	false																		
Format	NTFS																		
Label	Data																		
Letter	D																		
Order	2																		
PartitionID	2																		

Partition ID: 1 Active
Format: NTFS
Label: Vista_OS
Letter: C Order: 1

Partition ID: 2, Format: NTFS, Label: Data, Letter: D, Order: 2

1 logical drive, 100% of space. 0% unallocated.

Disk 0, 2 Partitions, 100% of space, 0% unallocated space.

Avoid tripping into "Extend trap"

You may have asked yourself what the Extend setting inside ModifyPartition is for. It's simple: It can only be used, if you've space left where the relevant partitioning object could be extended to. We have allocated the whole disk space using CreatePartition now. So there is no space left, where one of the ModifyPartition objects could be extended to.

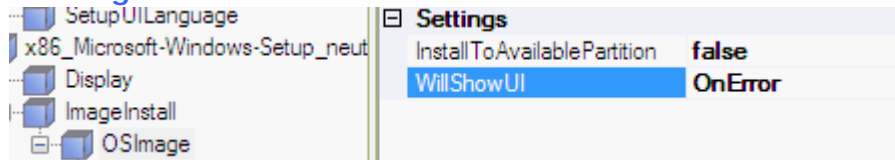
Partitioning Rule 6

Do **NOT** extend a partitioning object, which was **already extended** before in the same answer file!

Configuring InstallTo & InstallFrom

The **OSImage** component **is a must** to let run setup full unattended! First **InstallFrom\MetaData** will **point** setup to the right **Vista version**. Second **InstallTo** configures what the **target drive** and **target partition** is.

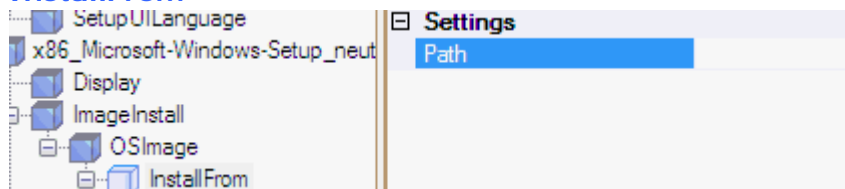
OSImage



If we would set up **InstallToAvailablePartition** on true, setup will choose the first available partition where no Windows OS is installed and which holds enough space.

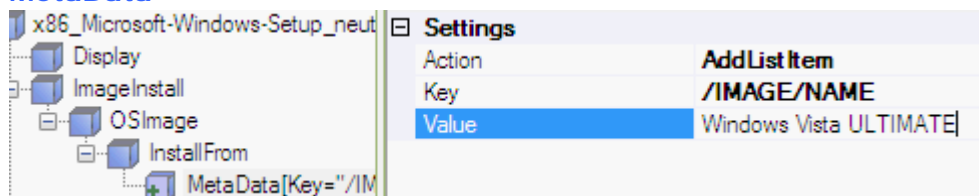
I highly recommend **NOT** to use **InstallToAvailablePartition** on **true**. We should **better use** **InstallTo** property to set up where to install Vista to exactly!

InstallFrom



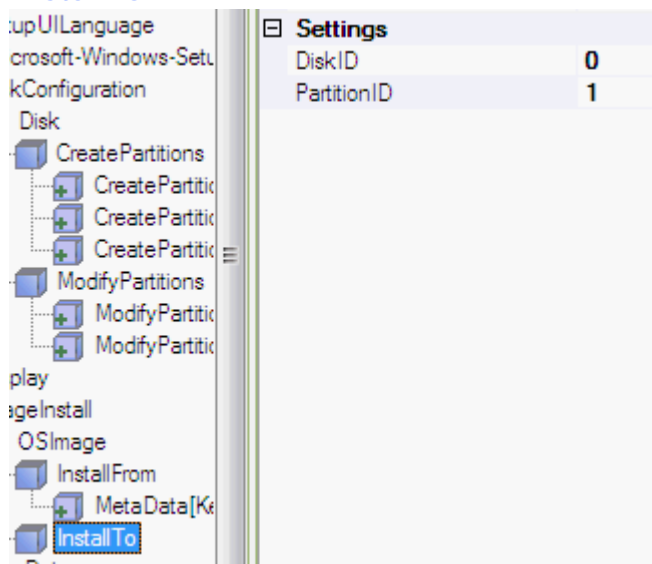
If we install from DVD or USB hard drive\stick we keep the path field empty! If we install from network share we put in UNC path here.

MetaData



[Check out here](#), for **more detailed informations** about MetaData.

InstallTo



If we use **InstallTo** to choose the partition, **InstallToAvailablePartition** must set to **false**. So you can't use **InstallToAvailablePartition** on true and **InstallTo** at the same time!

Partitioning Rule 7

Use **InstallTo** OR **InstallToAvailablePartition** on true, to set up the partition where Vista will be installed to. But do **NOT** use both settings at the same time.

Limitations

The last settings have already shown the limitations of setup internal partitioning: **You need to know the ID** of your target partition. That is still easy as long as we just use one hard drive.

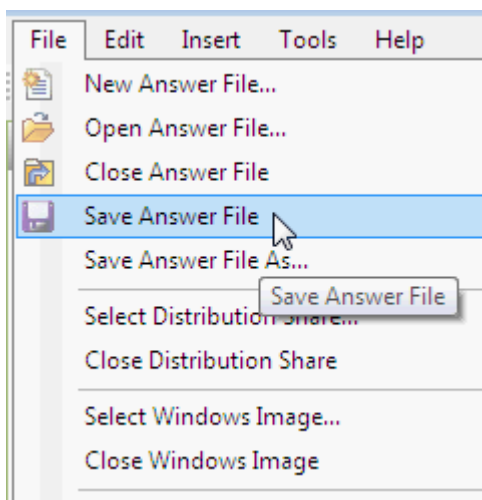
But if we use more than one disk - one on IDE controller and one on SATA controller for example - than it can be very difficult to figure out and to be sure which one the right ID is during PE setup pass.

In such cases I highly recommend NOT to use setup internal partitioning. We should better run our own, more flexible cmd script during PE pass or should do the partitioning manually than!

Partitioning Rule 8

Use setup internal partitioning carefully and use it only if **you're sure about the ID** of the target partition.

We choose **Save Answer File** from **File** menu



and we are ready to use the Autounattend.xml for setup.

Summary

Partitioning Rule 1

Ensure to make a **backup** of your **data before starting** any **automated partitioning** operations.

Partitioning Rule 2

One physical hard drive **MUST** have at least one primary partition.

Partitioning Rule 3

One physical drive can always have one active partition only.

Partitioning Rule 4

Extended partitions are **container objects** only. Create **logical drives inside** the extended partition **to let it become useable!**

Partitioning Rule 5

Never use Size AND Extend setting for a partitioning object at the same time! Use Size OR Extend.

Partitioning Rule 6

Do NOT extend a partitioning object, which was already extended before in the same answer file!

Partitioning Rule 7

Use InstallTo OR InstallToAvailablePartition on true, to set up the partition where Vista will be installed to. But do **NOT** use **both settings** at the **same time**.

Partitioning Rule 8

Use setup internal partitioning carefully and use it only if **you're sure about the ID** of the target partition.

Notes for WDS installations

If you want to use partitioning for WDS installations, than you have to place the partitioning settings into WDSClientUnattend.xml!

Partitioning XML Snippets

Here are some xml snippets for the most common partitioning scenarios.

WARNING:

Using these snippets will cause data lost. So please make data backups before using them!

1. Formatting first, already existing partition

This DiscConfiguration snippet will **format** the **first and already existing partition** inside our system:

```
<DiskConfiguration>
  <WillShowUI>OnError</WillShowUI>
  <Disk wcm:action="add">
    <ModifyPartitions>
      <ModifyPartition wcm:action="add">
        <Extend>false</Extend>
        <Format>NTFS</Format>
        <Label>Vista_OS</Label>
        <Letter>C</Letter>
        <Order>1</Order>
        <PartitionID>1</PartitionID>
        <Active>true</Active>
      </ModifyPartition>
    </ModifyPartitions>
    <DiskID>0</DiskID>
    <WillWipeDisk>false</WillWipeDisk>
  </Disk>
</DiskConfiguration>
```

2. Delete all partitions and create a new one

This DiskConfiguration snippet will **delete all existing partitions** on our first hard drive inside system and creates one new **primary partition**, which uses the **whole disk space**:

```
<DiskConfiguration>
  <Disk wcm:action="add">
    <CreatePartitions>
      <CreatePartition wcm:action="add">
        <Order>1</Order>
        <Extend>true</Extend>
        <Type>Primary</Type>
      </CreatePartition>
    </CreatePartitions>
    <ModifyPartitions>
      <ModifyPartition wcm:action="add">
        <Active>true</Active>
        <Extend>false</Extend>
        <Format>NTFS</Format>
        <Label>Vista-OS</Label>
        <Letter>C</Letter>
        <Order>1</Order>
        <PartitionID>1</PartitionID>
      </ModifyPartition>
    </ModifyPartitions>
    <DiskID>0</DiskID>
    <WillWipeDisk>true</WillWipeDisk>
  </Disk>
  <WillShowUI>OnError</WillShowUI>
</DiskConfiguration>
```

3. Delete all partitions and create two new partitions

This DiskConfiguration snippet will **delete all existing partitions** on our first hard drive inside system and creates one new **primary partition (25 GB, Vista_OS)** and a second **data partition (Data)** , which uses the **rest of disk space**:

```
<DiskConfiguration>
  <Disk wcm:action="add">
    <CreatePartitions>
      <CreatePartition wcm:action="add">
        <Order>1</Order>
        <Size>25000</Size>
        <Type>Primary</Type>
      </CreatePartition>
      <CreatePartition wcm:action="add">
        <Extend>true</Extend>
        <Order>2</Order>
        <Type>Extended</Type>
      </CreatePartition>
      <CreatePartition wcm:action="add">
        <Order>3</Order>
        <Type>Logical</Type>
        <Extend>true</Extend>
      </CreatePartition>
    </CreatePartitions>
    <ModifyPartitions>
      <ModifyPartition wcm:action="add">
        <Active>true</Active>
        <Extend>false</Extend>
        <Format>NTFS</Format>
        <Label>Vista-OS</Label>
        <Letter>C</Letter>
        <Order>1</Order>
        <PartitionID>1</PartitionID>
      </ModifyPartition>
      <ModifyPartition wcm:action="add">
        <Active>false</Active>
        <Extend>false</Extend>
        <Format>NTFS</Format>
        <Label>Data</Label>
        <Letter>D</Letter>
        <Order>2</Order>
        <PartitionID>2</PartitionID>
      </ModifyPartition>
    </ModifyPartitions>
    <DiskID>0</DiskID>
    <WillWipeDisk>true</WillWipeDisk>
  </Disk>
  <WillShowUI>OnError</WillShowUI>
</DiskConfiguration>
```

4. Delete all partitions and create three new partitions

This DiskConfiguration snippet will **delete all existing partitions** on our first hard drive inside system and creates one **new primary partition (25 GB, Vista_OS)**, a **second programs partition (40 GB, Programs)**, and a **third data partition (Data)**, which uses the **rest of disk space**:

```
<DiskConfiguration>
  <Disk wcm:action="add">
    <CreatePartitions>
      <CreatePartition wcm:action="add">
        <Order>1</Order>
        <Size>25000</Size>
        <Type>Primary</Type>
      </CreatePartition>
      <CreatePartition wcm:action="add">
        <Extend>true</Extend>
        <Order>2</Order>
        <Type>Extended</Type>
      </CreatePartition>
      <CreatePartition wcm:action="add">
        <Extend>false</Extend>
        <Order>3</Order>
        <Type>Logical</Type>
        <Size>40000</Size>
      </CreatePartition>
      <CreatePartition wcm:action="add">
        <Size>10</Size>
        <Order>4</Order>
        <Type>Logical</Type>
      </CreatePartition>
    </CreatePartitions>
    <ModifyPartitions>
      <ModifyPartition wcm:action="add">
        <Active>true</Active>
        <Extend>false</Extend>
        <Format>NTFS</Format>
        <Label>Vista_OS</Label>
        <Letter>C</Letter>
        <Order>1</Order>
        <PartitionID>1</PartitionID>
      </ModifyPartition>
      <ModifyPartition wcm:action="add">
        <Active>false</Active>
        <Extend>false</Extend>
        <Format>NTFS</Format>
        <Label>Programs</Label>
        <Letter>D</Letter>
        <Order>2</Order>
        <PartitionID>2</PartitionID>
      </ModifyPartition>
      <ModifyPartition wcm:action="add">
        <PartitionID>3</PartitionID>
        <Order>3</Order>
        <Label>Data</Label>
        <Format>NTFS</Format>
        <Active>false</Active>
        <Extend>true</Extend>
        <Letter>E</Letter>
      </ModifyPartition>
    </ModifyPartitions>
    <DiskID>0</DiskID>
    <WillWipeDisk>true</WillWipeDisk>
  </Disk>
</DiskConfiguration>
```

EXPLANATION:

This last example does **use a workaround** to create a second logical drive inside extended partition.

It seems to be that you can use Extend tag for the first partition inside the extended partition only.

If you want to let allocate the rest of space inside extended partition to a second logical drive, you've to set a size for the second logical drive using create partition and then you can extend it to the rest of space using modify partition.

If you use extend without setting a size before, the second logical drive will not be created.

[Check out here](#), too.

HowTo: Creating an iso file of Vista DVD

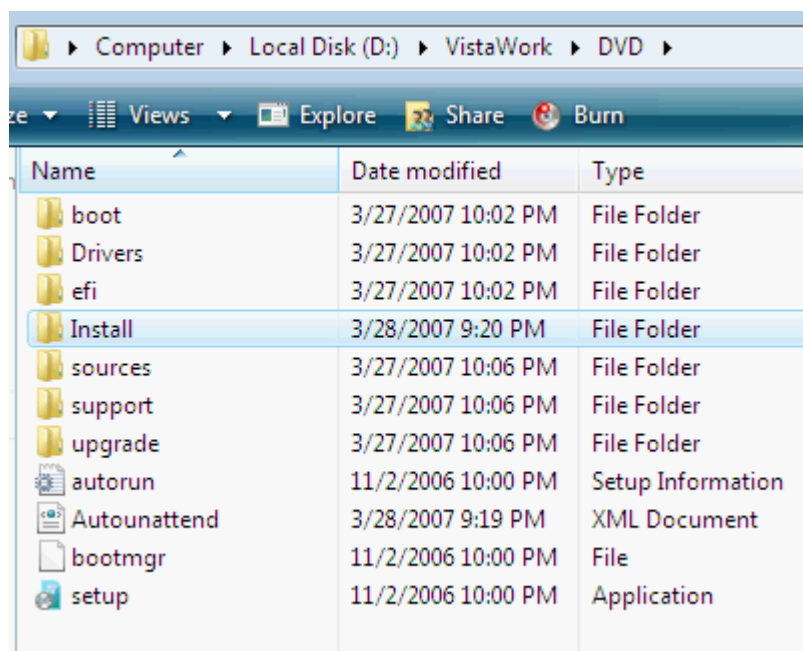
What we need

1. **WAIK** installed
2. **Vista DVD** copied to a folder on HD

Step 1: Putting all files and folders into one directory on HD

First of all we **copy** the **Vista DVD** to a folder on our **HD** (eg.: **D:\VistaWork\DVD**).

Now we add **additional stuff** like **Driver** folder or **AutoUnattend.xml**. So we will get following structure:



Name	Date modified	Type
boot	3/27/2007 10:02 PM	File Folder
Drivers	3/27/2007 10:02 PM	File Folder
efi	3/27/2007 10:02 PM	File Folder
Install	3/28/2007 9:20 PM	File Folder
sources	3/27/2007 10:06 PM	File Folder
support	3/27/2007 10:06 PM	File Folder
upgrade	3/27/2007 10:06 PM	File Folder
autorun	11/2/2006 10:00 PM	Setup Information
Autounattend	3/28/2007 9:19 PM	XML Document
bootmgr	11/2/2006 10:00 PM	File
setup	11/2/2006 10:00 PM	Application

Step 2: Create ISO file

We open **Start\Program Files\Microsoft Windows AIK\Windows PE Tools Command Prompt** and insert

```
cd
```

now we change to **PETools directory** typing

```
cd PETools
```

If we want to create an iso file of **32 bit Vista**, we type:

```
oscdimg -n -m -b"%Programfiles%\Windows AIK\Tools\PETools\x86\boot\etfsboot.com" D:\VistaWork\DVD  
D:\VistaWork\ISO\Vistax86.iso
```

If we want to create an iso file of **64 bit Vista**, we type:

```
oscdimg -n -m -b"%Programfiles%\Windows AIK\Tools\PETools\amd64\boot\etfsboot.com" D:\VistaWork\DVD  
D:\VistaWork\ISO\Vista64.iso
```

EXPLANATION:

```
oscdimg -n -m -b"%Programfiles%\Windows AIK\Tools\PETools\x86\boot\etfsboot.com" D:\VistaWork\DVD  
D:\VistaWork\ISO\Vistax86.iso
```

-b points to the location of the boot sector file (**red part**). This file will make the DVD bootable. **Do not use a space between b and path!**

The **blue part** points to the location of Vista DVD files.

The **purple part** points to the directory, where the iso file will be created in.

-n makes it possible to use extended file names.

-m allows to create iso files, larger than CD format.

Step 3: Burn ISO onto DVD media using third party software

Now we need to burn the iso file onto DVD using a third party software, like Nero or **ImageBurn**.

Known issue: Autounattend.xml will not to be executed

There were a few threads coming up on MSFN and other forums, discussing where to place Autounattend.xml on DVD. Most people say, it should be placed into the root of DVD, others were using it successfully inside sources folder of DVD.

Vista setup follows a special order, to look for Autounattend.xml described in the document ["Methods for Running Windows Setup"](#) on MS TechNet.

If you have problems getting your Autounattend.xml detected on the root, put it to the sources folder. One or the other alternate supposed to work on your system. I have not found about a rule of this behavior yet. Seems to be, that VMs like having it inside sources.

ADVANCED

HowTo: Injecting drivers into Vista image (install.wim)

NOTE:

This method will **NOT** work for driver packages like .exe or .msi files. It will work with inf files only!

Introduction

Vista Setup offers **different methods to inject drivers**. This article describes the **offline injection** of drivers. "**Offline**" means the **drivers are injected directly into** the Vista Setup source - the **install.wim**. This is done in a **separate procedure before setup applies the image** to your hard drive.

IMPORTANT:

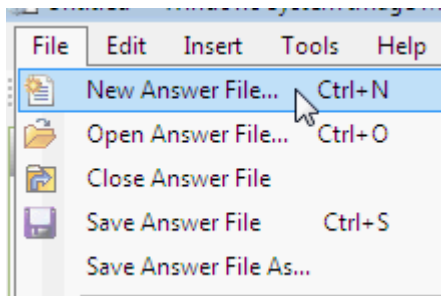
If you want to inject new drivers to your install.wim you have to start with a "clean" install.wim, where no drivers were injected before. Injecting drivers to an install.wim, where you were using this method before can cause problems!!!

What we need

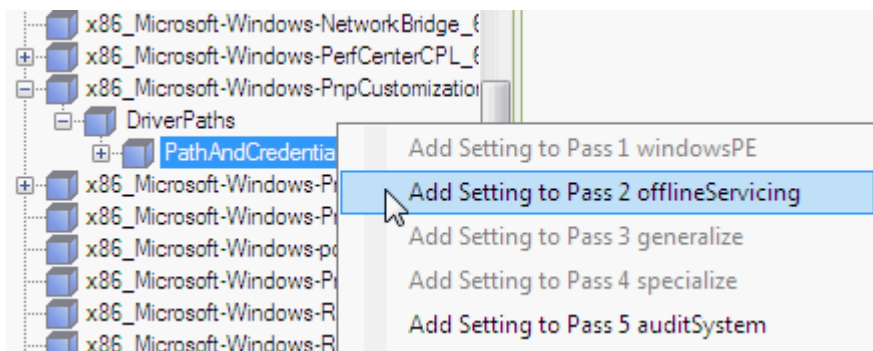
1. Vista DVD copied to folder on HD
2. Microsoft's WAIK

Step 1: Create new unattend.xml using WSIM

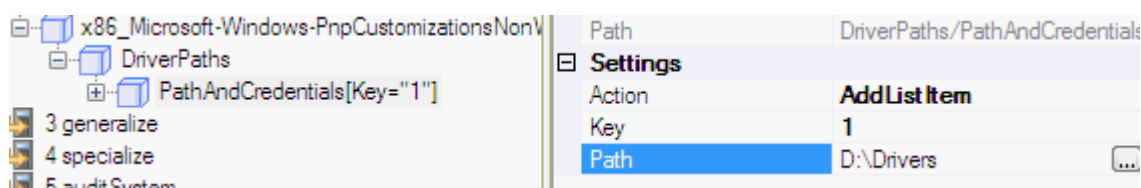
We **start WSIM** and choose **New Answer File...** from **File** menu.



From **Windows Image** pane we **right click** the component **Microsoft-Windows-PnpCustomizationsNonWinPE \ DriverPath \ PathandCredentials**, and **add** it to **offlineServicing** pass of our answer file.



In Answer File pane we **left click** the new **PathAndCredentials** and do the following settings in **Settings** pane:

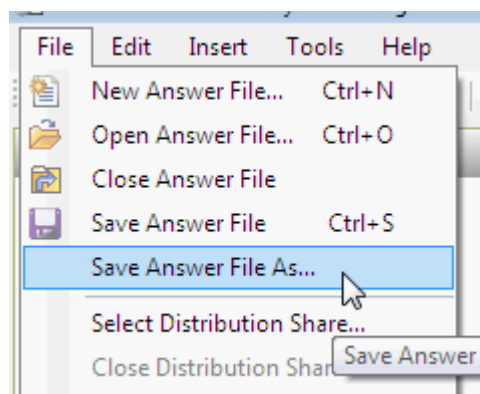


D:\Drivers you have to replace with the folder, where you hold your inf driver files.

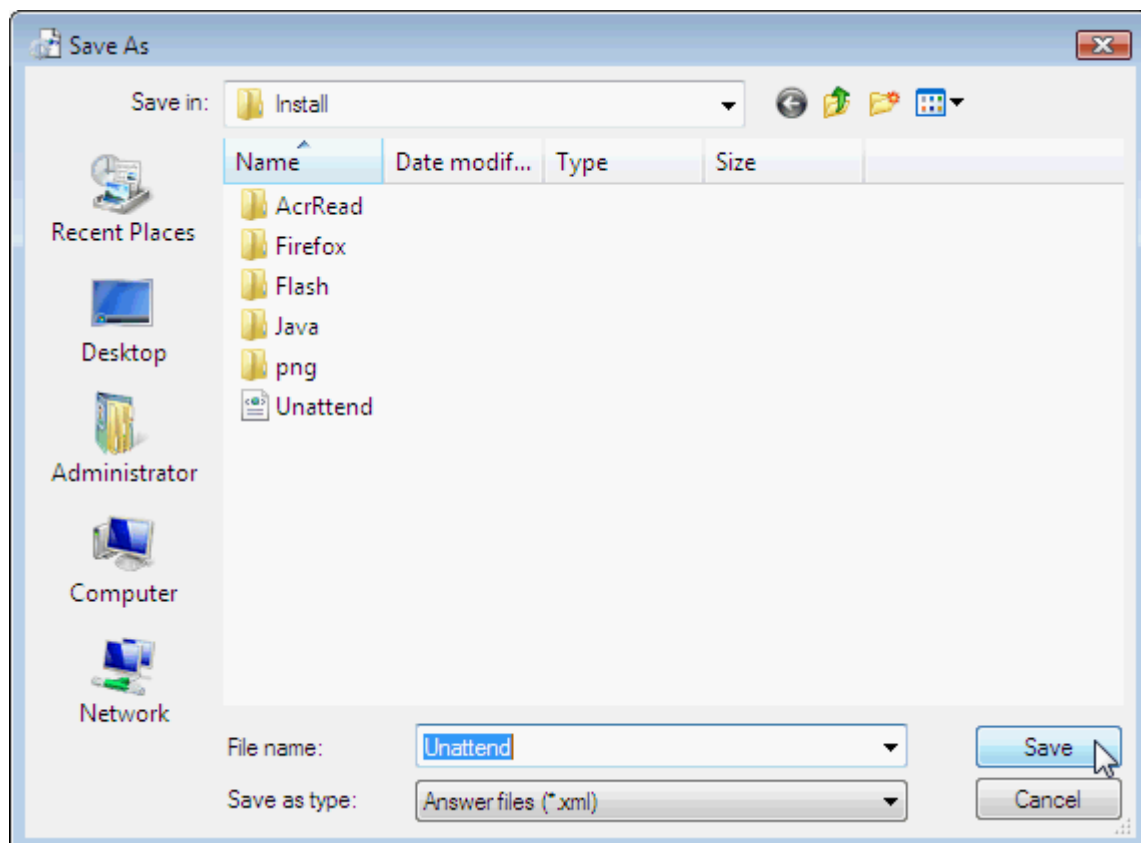
NOTE:

Credentials settings are used only, when your driver folder is located on a network share. If you're holding them on the same system, keep these settings empty!

Form **File** menu we choose **Save Answer File as...**



and save the file as **unattend.xml**.



Now we can close **WSIM**.

NOTE:

This **unattend.xml** will be used for the driver injection only. It has nothing to do with **Autounattend.xml** we will use later during **Vista Setup**.

Step 2: Mount install.wim



We click on **Start\Program Files\Microsoft Windows AIK\Windows PE Tools Command Prompt** .
Type the following command into the appearing cmd window:

```
imagex /mountrw D:\VistaWork\DVD\sources\install.wim 4 D:\VistaWork\Mount
```

EXPLANATION:

`imagex /mountrw D:\VistaWork\DVD\sources\install.wim 4 D:\VistaWork\Mount`

The **red part** represents the path to `install.wim`.

The **blue 4** identifies one of the image index IDs within `install.wim`.

Explanation:

The `install.wim` contains seven "sub-images". For every Vista version exists one "sub-image". 4 is equivalent to Vista Ultimate.

To find out which ID belongs to which Vista version, type

```
imagex /info D:\VistaWork\DVD\sources\install.wim
```

into cmd-box.

The **purple part** represents the path where the `install.wim` is mounted to.

/mountrw opens the image for **read and write** operations.

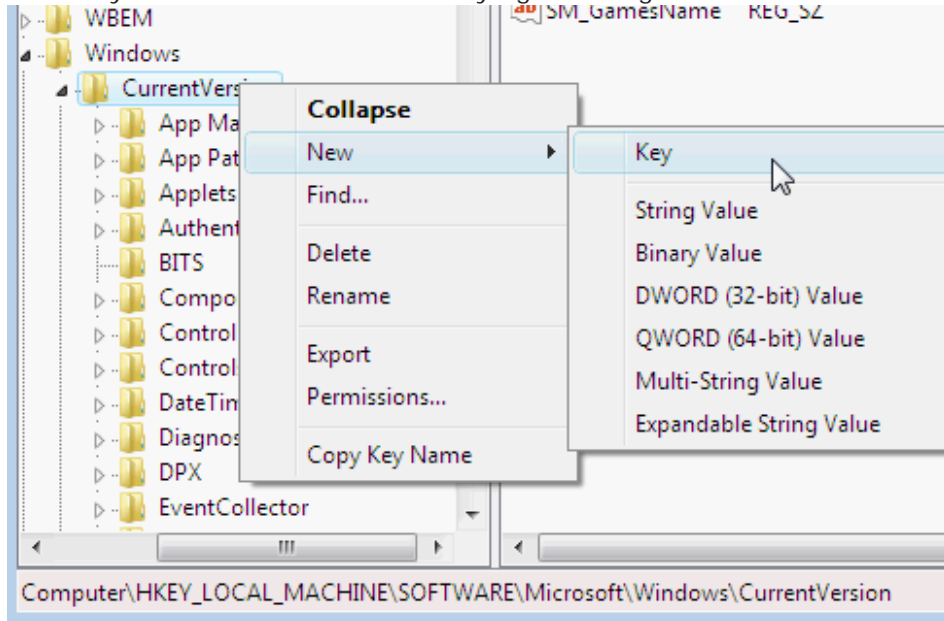
If you want to open the image for reading only, you can use **/mount**. It's not possible to save any changes to the image than!

Step 3: Activating advanced log function for driver installations

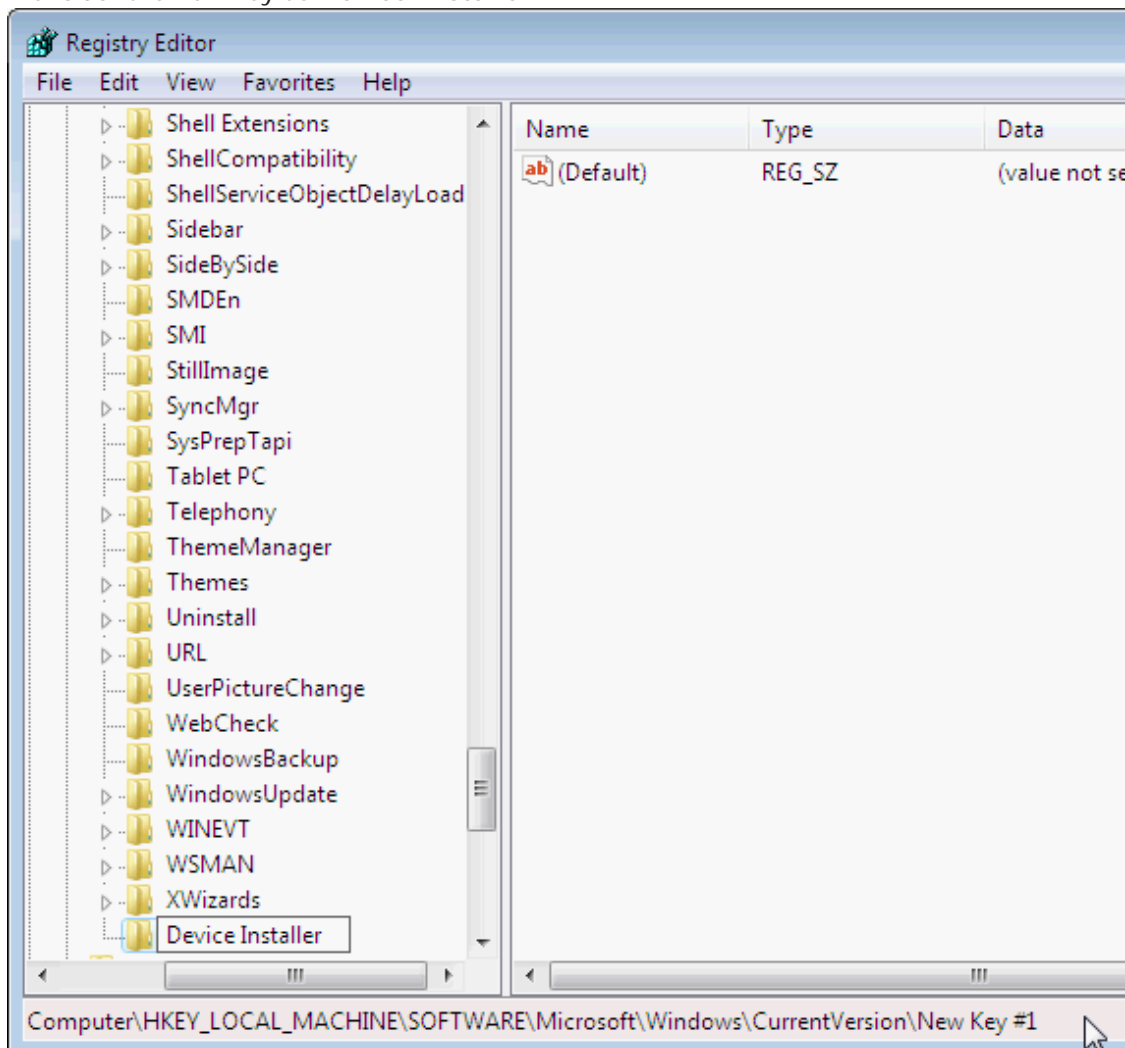
We click **Start** and **Run** and type **regedit** and hit enter. Now we extend the view - by clicking the plus - till we reach the following key:

HKLM\Software\Microsoft\Windows\CurrentVersion\Device Installer

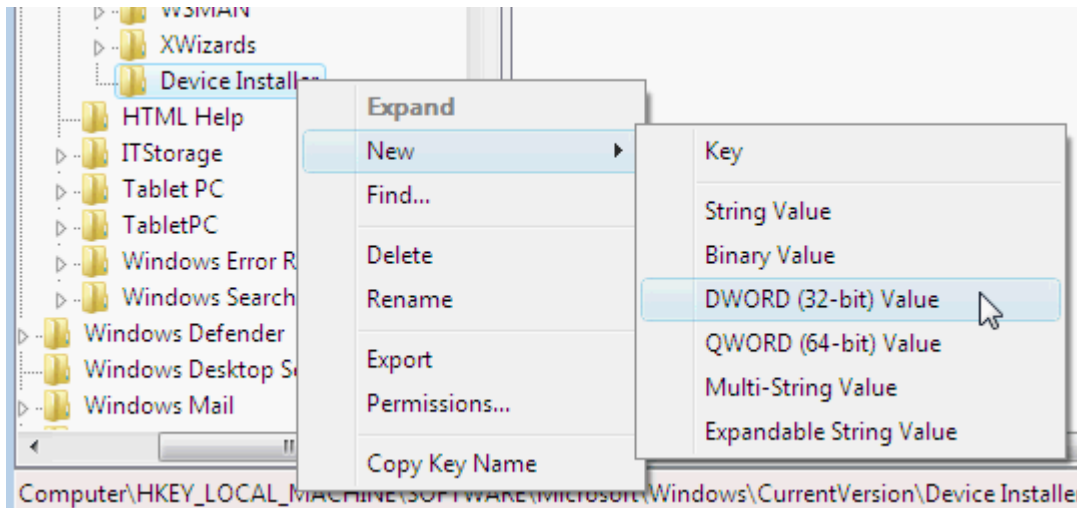
If the key does not exist we create it by right clicking on **CurrentVersion** and choose **New \ Key**:



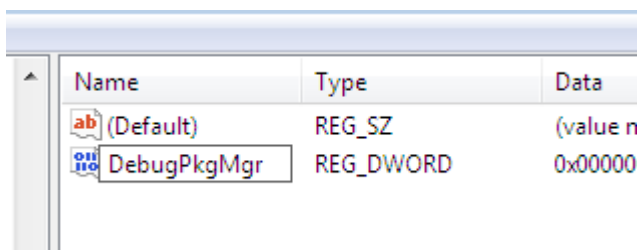
We **label** the new key as **Device Installer**:



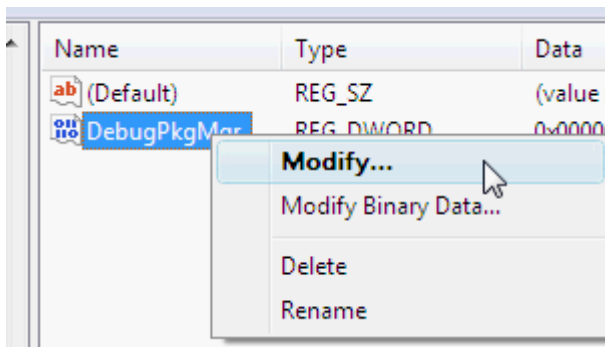
We right click on the new Key **Device Installer** and choose **New \ DWord**:



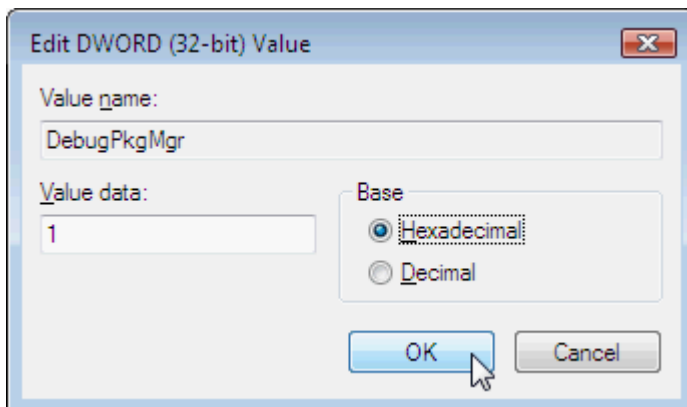
We **label** the new key as **DebugPkgMgr**:



We **right click** **DebugPkgMgr** and choose **Modify**:



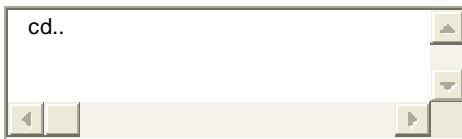
We insert **1** for **Value data** and leave **Base** on **Hexadecimal** and hit **OK**:



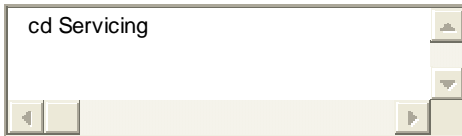
We are **finished** and can **close Regedit**.

Step 4: Execute Packagemanager

Insert the following command into **PE Tools Command Prompt**:



With



we browse to **Servicing folder**.

Now we type the following:

```
pkgmgr /o:"D:\VistaWork\mount\;D:\VistaWork\mount\Windows" /n:"D:\VistaWork\xml\unattend.xml"  
/l:"D:\VistaWork\Logs\inject.log"
```

EXPLANATION:

```
pkgmgr /o:"D:\VistaWork\mount\;D:\VistaWork\mount\Windows" /n:"D:\VistaWork\xml\unattend.xml"  
/l:"D:\VistaWork\Logs\inject.log"
```

The **red part** points to the path where the install.wim is mounted to and the Windows directory is located.

The **blue part** includes the path to the answer file (unattend.xml), we have created before.

The **purple part** points to the location of the log file which is created during injection procedure.

After package manager has finished, we can **open inject.log.txt** to **check if the injection** was ended **successfully**. If everything was fine the log ends with the following line:

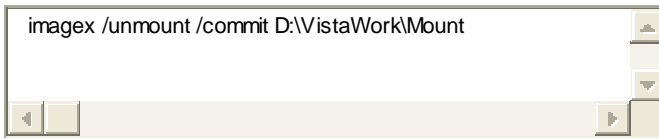
return code: 0x0

To **double-check** you can **use Explorer** to check, if drivers were integrated properly. We browse to the folder where we've mounted the image before – here **D:\VistaWork\Mount**.

We open **Windows\inf**. The driver files will be injected as **oemN.inf**, where **N** counts up starting from **0**.

Step 5: Unmount install.wim

If everything went fine, we insert the following command **into PE Tools Command Prompt**



This will write back the image to install.wim located in DVD folder – here

D:\VistaWork\DVD\sources\install.wim. This will **take a while**, so **do NOT cancel this operation!**

EXPLANATION:

/unmount is closing the wim image file.

/commit is saving all changes. **Without using commit all changes will be lost!**

Step 6: Create ISO file

Create an ISO file following [steps described here](#), burn it to DVD and you're ready to roll out the Vista-DVD with your drivers injected.

HowTo: Install drivers from DVD or USB flash drive directly

Introduction

This article describes, how to **install drivers** during unattended Vista Setup **directly from removable media** like USB flash drive, DVD etc. It **avoids copying all content of** your removable **media to HD** like `<UseConfigurationSet>` tag does.

NOTE:

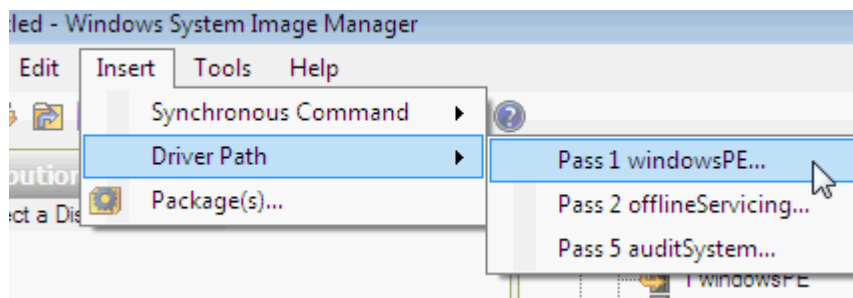
This **method** will **NOT work** for **driver packages** like **.exe** or **.msi** files. It will work with **inf files only!**

What we need

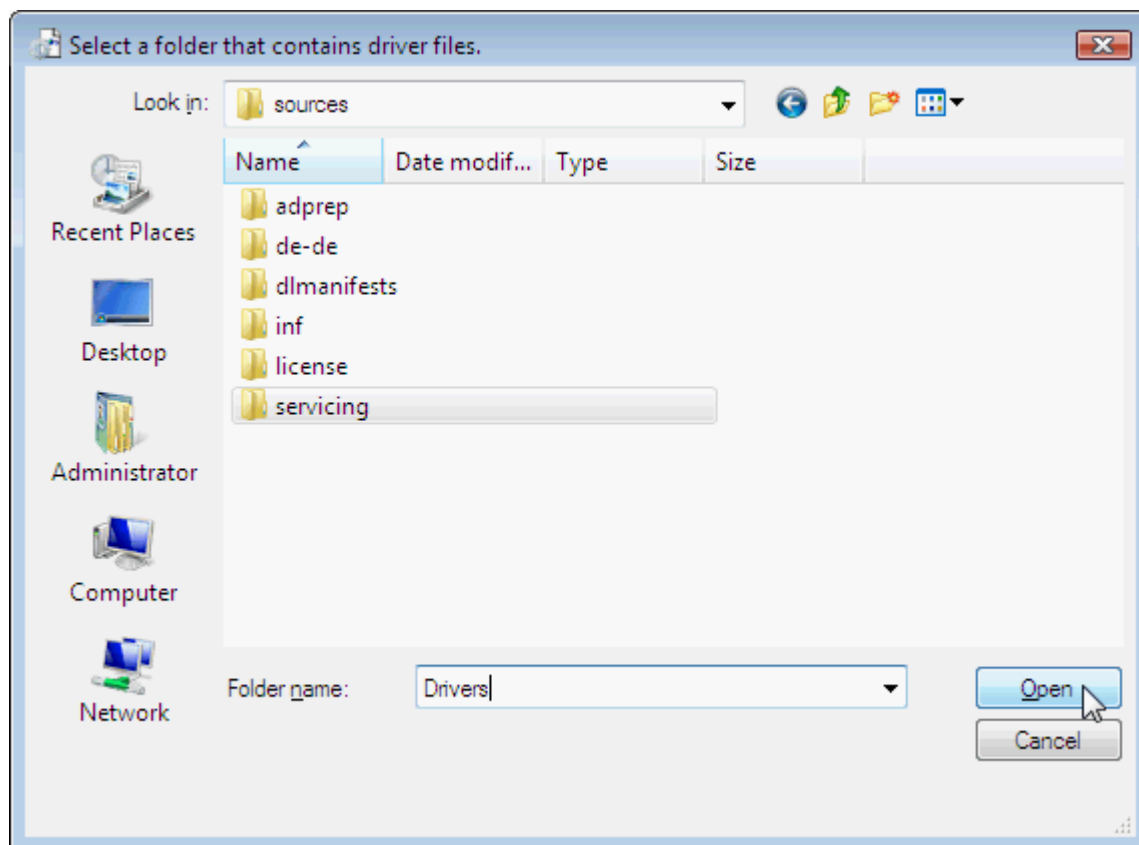
1. **Vista-DVD**, copied to folder on HD
2. **WAIK** installed
3. **Basic answer file (Autounattend.xml)**
4. **Additional Drivers (inf format)**

Step 1: Add Driver Path to Windows PE pass of your Autounattend.xml

From **Insert** menu we choose **Driver Path** and **Pass 1 windowsPE...**

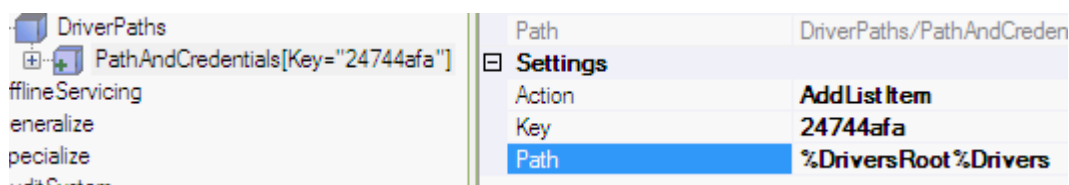


In the next window we'll be **prompted to choose a directory**.



It **doesn't matter what we're choosing** here, we need to change it anyway.

You will now see an entry called **Microsoft-Windows-PnPCustomizationsWinPE \ DriverPaths \ PathAndCredentials** in Answer File pane. We **left click it** and do the following changes for **Path** setting:



NOTE:

Your **Key** setting will **may look different** from the one above, but that **doesn't matter**. Just **leave the Key setting** like it is.

We **don't need** to do set anything for **Credentials**, **cause** our **drivers** will be located **on removable media**.

Step 2: Mounting boot.wim

Next we will **mount** the **boot.wim** to do some editing. First of all we **create a folder** we can mount the boot.wim to (eg. **D:\VistaWork\Mount**).

We choose **Start \ Program Files \ Microsoft Windows AIK \ the Windows PE Tools Command Prompt** and insert the following command:

```
imagex /mountrw D:\VistaWork\DVD\sources\boot.wim 2 D:\VistaWork\Mount
```

After that we are able to **browse and edit** the **contents** of boot.wim in **D:\VistaWork\Mount**.

Step 3: Creating winpeshl.ini

We browse to **D:\VistaWork\Mount\Windows\system32** and **right click** on **empty space**. From appearing **context menu** we choose **New\Text Document** and copy the following lines to that new file:

```
[LaunchApp]
AppPath=%Systemdrive%\Scripts\SetDriversRoot.cmd
```

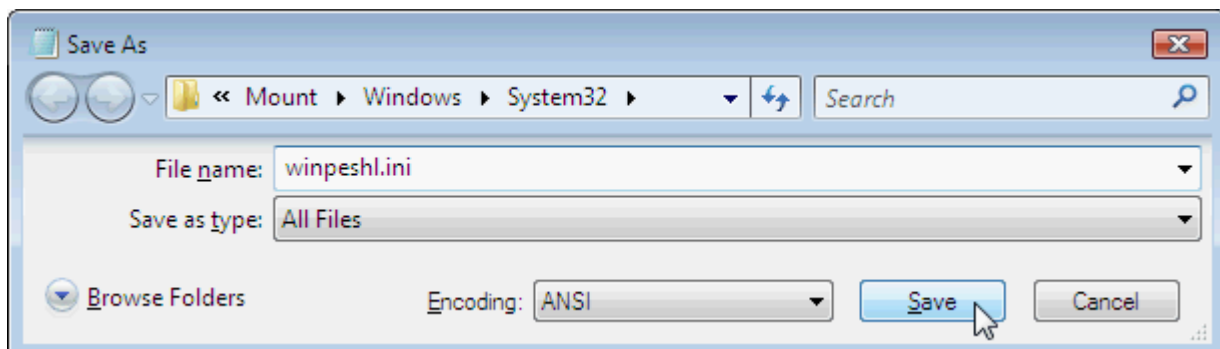
IMPORTANT:

The **application** we're **starting** using winpeshl.ini is **treated as** the **PE shell**. That means it will **keep Windows PE running** OR it can **exit Windows PE** and cause a reboot!

So if we would exit the SetDriversRoot.cmd for example, Windows PE would exit immediately!

That's important to know like we will see later.

From **File menu** we choose **Save as...** and save the file as **winpeshl.ini**. Make sure, that **All Files** is chosen as **Save as type**.



Step 4: Creating the SetDirversRoot.cmd

We browse to **D:\VistaWork\Mount** and **right click** on **empty space**. From appearing **context menu** we choose **New\Folder** and **label** this folder as **Scripts**.

We **open** the new **Scripts** folder **right click** on **empty space** again and choose from **context menu** **New\Text Document**. We copy the following lines to that new text file:

```
@ECHO OFF
SET DriversRoot=NULL

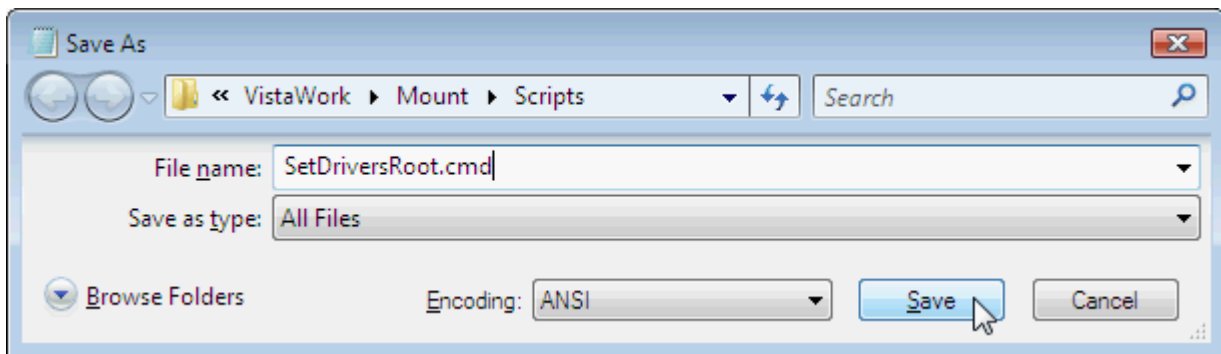
:SearchDriversRoot
FOR %%i IN (C D E F G H I J K L M N O P Q R S T U V W X Y Z) DO IF EXIST %%i:\DriversRoot.txt SET
DriversRoot=%%i:\
IF NOT "%DriversRoot%"=="NULL" GOTO StartSetup
GOTO SearchDriversRoot

:StartSetup
ECHO DriversRoot=%DriversRoot%
X:\setup.exe
```

IMPORTANT:

It's **important** that we **do NOT** use an **EXIT** command or **EOF** at the end of this file. **Otherwise** Windows PE would exit before setup is started!

From **File** menu we choose **Save as...** and save the file as **SetDriversRoot.cmd**. Make sure again, that **Save as type** is set to **All Files**:



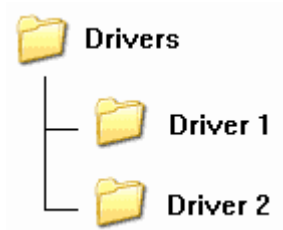
Step 5: Unmount boot.wim saving changes

We open **Start \ Program Files \ Microsoft Windows AIK \ Windows PE Tools Command Prompt** and use the **following command** to **unmount** the boot.wim and **save the changes** we have done before:



Step 6: Creating drivers directory

We **create a new folder** and **label it Drivers**. Now we **copy** all of our **additional inf driver files** to that new **Drivers** folder. **You can create subfolders** if you want but you don't have to. By creating **subfolders** you **influence the order** the **drivers will be installed** later during Setup:

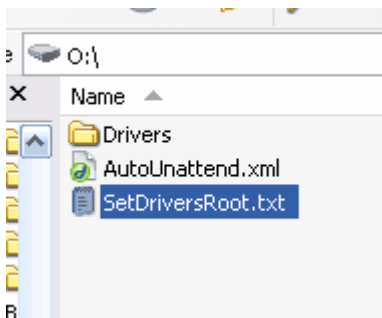


Step 7: Copying files and folder to removable media

Option 1: Copying to USB flash drive

We **copy the Drivers** folder and the adapted **AutoUnattend.xml** to the **root** of our **USB flash drive**.

At least we have to create a **new text file** on the **root of USB flash drive** called **DriversRoot.txt**. All this will **result** in the following **structure**:

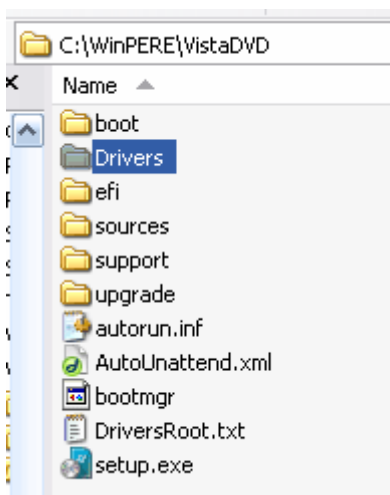


Now the USB flash drive is ready to use. Just put it together with the Setup DVD into your system and let boot the machine.

Option 2: Copying to DVD

We **copy the Drivers** folder and the adapted **AutoUnattend.xml** to the **root** of our **Vista DVD** folder.

At least we have to create a **new text file** on the **root of USB flash drive** called **DriversRoot.txt**. All this will **result** in the following **structure**:



We open **Start \ Program Files \ Microsoft Windows AIK \ Windows PE Tools Command Prompt** and use the following command to **create** an **ISO file** of this structure:

```
oscdimg -n -m -bD:\VistaWork\DVD\boot\etfsboot.com D:\VistaWork\DVD D:\VistaWork\ISO\Vista.iso<br>
```

A file called **Vista.iso** will be created in **D:\VistaWork\ISO**. Use a burning software to **burn the ISO file to a DVD** and you're **ready to use** the DVD for Vista Setup.

IMPORTANT:

Do **NOT** use both alternatives at the same time!!!

EXPLANATION:

At first **Vista DVD boots to Windows PE**. Windows PE is the contents of boot.wim. **During PE startup** sequence the **winpeshl.ini** is executed automatically. The **winpeshl.ini** calls the **SetDriversRoot.cmd** we created before.

The **SetDriversRoot.cmd** search all drives on your system for the **DriversRoot.txt**. Once it has found the **DriversRoot.txt** it will set the **%DriversRoot%** variable with **drive letter** where **DriversRoot.txt** was located. Now the **%DriversRoot%** variable can be used in **Autounattend.xml** to point to the **Drivers** folder.

NOTE:

Using this method the **%DriversRoot%** variable is **present during PE pass only!** If you want to use the **%DriversRoot%** variable during a **later pass**, you could do it the same way, but you would **have to put the SetDriversRoot into install.wim** to be present in later passes.

FAQ

Q:

I was doing the same thing using the **<UseConfigurationSet>** setting and **\$OEM\$** folder structure and it was **working well and easier** so far. So what is the **benefit of the SetDriversRoot method?**

A:

It's right that you can install drivers easier using the **<UseConfigurationSet>** setting. But there is a big disadvantage of this method you should know about:

If you use the **<UseConfigurationSet>** setting in your **Autounattend.xml** the **WHOLE CONTENTS** of the **media** where **\$OEM\$** folder is located, will be **copied to %Windir%\ConfigSetRoot** of your Vista system. That means if your **\$OEM\$** folder is located on your Vista DVD the whole Vista DVD is copied to **%Windir%\ConfigSetRoot**. This is **wasting time and a lot of space!**

If you use the method described in this article this will not happen. Only driver files will be copied over to your Vista system.

Q:

Are there some **disadvantages** using **SetDriversRoot** method?

A:

Yes, there is. Using SetDriversRoot method ALL drivers **which are** present in **Drivers folder** will be copied **over** to **your** Vista system, **regardless if they are needed by the system or not.**

This is not a typical behavior of this method. **This is** caused by the way Setup handles drivers. **Once Vista setup finds drivers you pointed it to it will copy over all driver files it found.**

HowTo: Slipstreaming updates into Vista installation source (install.wim)

Introduction

There are several possibilities to integrate updates and hotfixes during Vista setup. This HowTo describes **copying (slipstreaming)** the **update files** right **into** Vista installation source - **install.wim**.

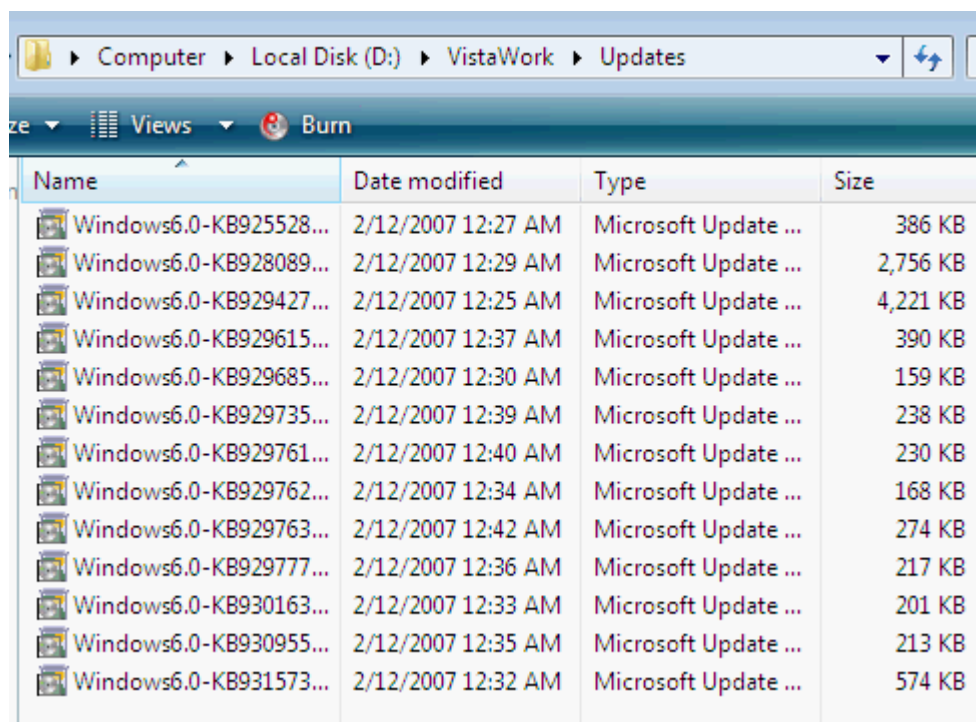
What we need

1. **WAIK** installed
2. **Working folder structure**

Alternative I - Manual Integration

Step 1: Copying updates into one folder

First of all we **copy** all downloaded **msu update files** into one folder - **D:\VistaWork\Updates**:



Name	Date modified	Type	Size
Windows6.0-KB925528...	2/12/2007 12:27 AM	Microsoft Update ...	386 KB
Windows6.0-KB928089...	2/12/2007 12:29 AM	Microsoft Update ...	2,756 KB
Windows6.0-KB929427...	2/12/2007 12:25 AM	Microsoft Update ...	4,221 KB
Windows6.0-KB929615...	2/12/2007 12:37 AM	Microsoft Update ...	390 KB
Windows6.0-KB929685...	2/12/2007 12:30 AM	Microsoft Update ...	159 KB
Windows6.0-KB929735...	2/12/2007 12:39 AM	Microsoft Update ...	238 KB
Windows6.0-KB929761...	2/12/2007 12:40 AM	Microsoft Update ...	230 KB
Windows6.0-KB929762...	2/12/2007 12:34 AM	Microsoft Update ...	168 KB
Windows6.0-KB929763...	2/12/2007 12:42 AM	Microsoft Update ...	274 KB
Windows6.0-KB929777...	2/12/2007 12:36 AM	Microsoft Update ...	217 KB
Windows6.0-KB930163...	2/12/2007 12:33 AM	Microsoft Update ...	201 KB
Windows6.0-KB930955...	2/12/2007 12:35 AM	Microsoft Update ...	213 KB
Windows6.0-KB931573...	2/12/2007 12:32 AM	Microsoft Update ...	574 KB

Step 2: Mounting install.wim

To integrate the updates, we need to mount install.wim at first. We open **Start\Program Files\Microsoft Windows AIK \Windows PE Tools Command Prompt** to insert the following command:

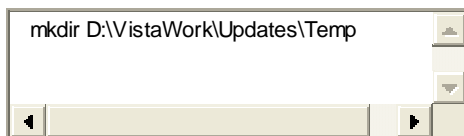
```
imagex /mountrw D:\VistaWork\DVD\sources\install.wim 4 D:\VistaWork\Mount
```

NOTE:

This will mount the ULTIMATE version from original Vista DVD. If you want to mount an other version, you may have to insert something else. Check out [here](#) under for more detailed information.

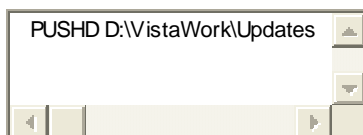
Step 3: Creating temporary folder

Next we create a **temporary folder**, where we will extract the msu files to:



Step 4: Extracting update files

We go back to **Command Prompt** and jump to the Updates directory:



We will now **extract** the **msu files** using **Expand** inside an **FOR loop** to do it **for all update files** inside our Updates folder:

```
FOR %i IN (*) DO (START "Expand" /WAIT "%Programfiles%\Windows AIK\Tools\Serviceicing"\Expand.exe %i  
-f:* D:\VistaWork\Updates\Temp)
```

The **updates** will be **extracted to** the **Temp** folder **inside Updates folder**.

NOTE:

The **command line above** is for use under **Windows XP**. If you're using **Vista**, **Expand.exe** is included already and located in **%Windir%\System32**. So the **command line for Vista** has to look like this:

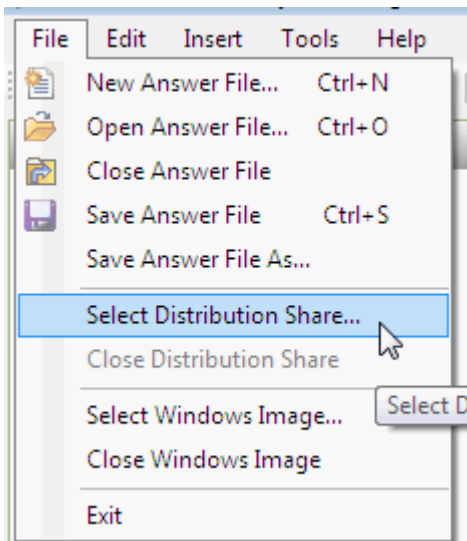
```
FOR %i IN (*) DO (START "Expand" /WAIT %Windir%\System32\Expand.exe %i -f:* D:\VistaWork\Updates\Temp)
```

IMPORTANT:

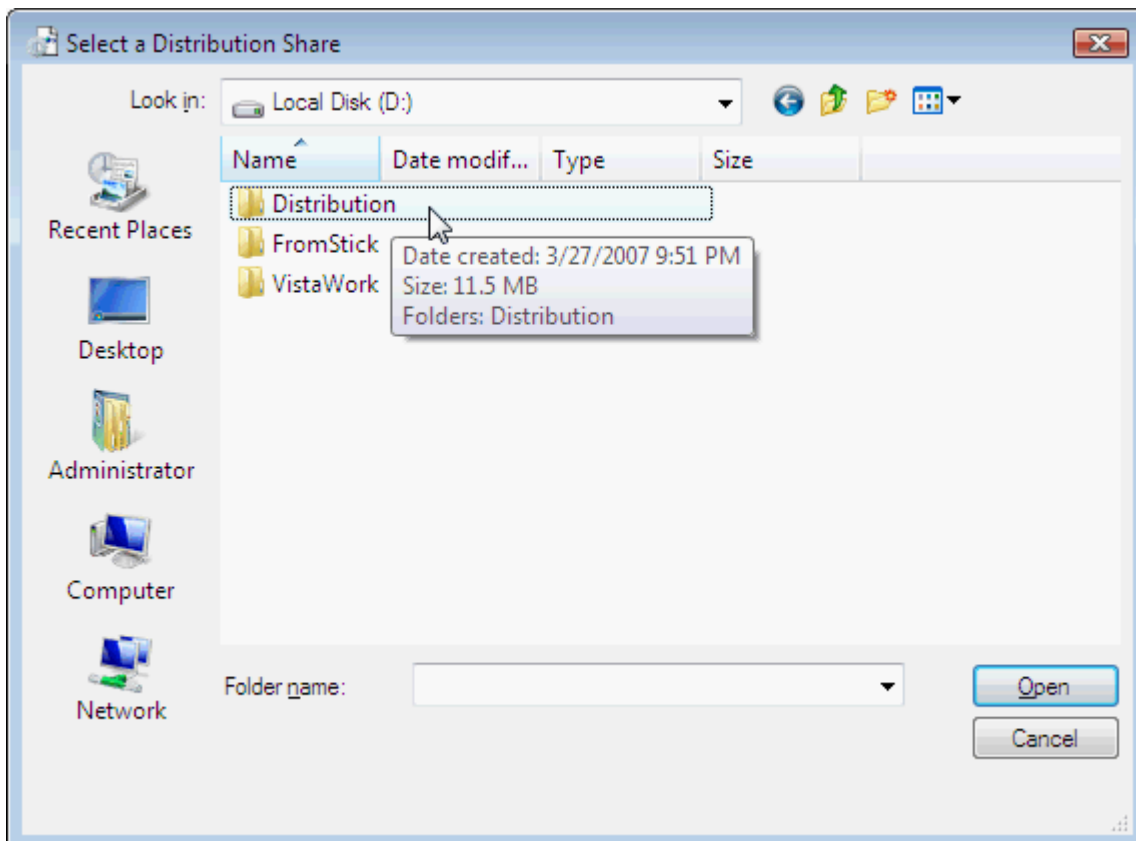
The **integration can take a while**. During the **execution command windows** will open and close automatically per update. **Do NOT cancel this process!**

Step 5: Importing update packages to WSIM

Next we need to **select a Distribution Share** to be able to import the update cab files. So we **open WSIM** and **Select Distribution Share...** from **File** menu:

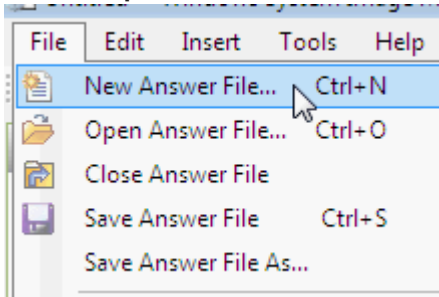


If we **do have installed BDD 2007** we **should use** the **Distribution Share** we use with BDD:

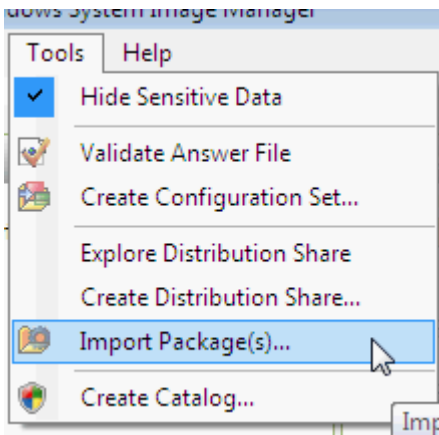


If not we can **create any new folder** here.

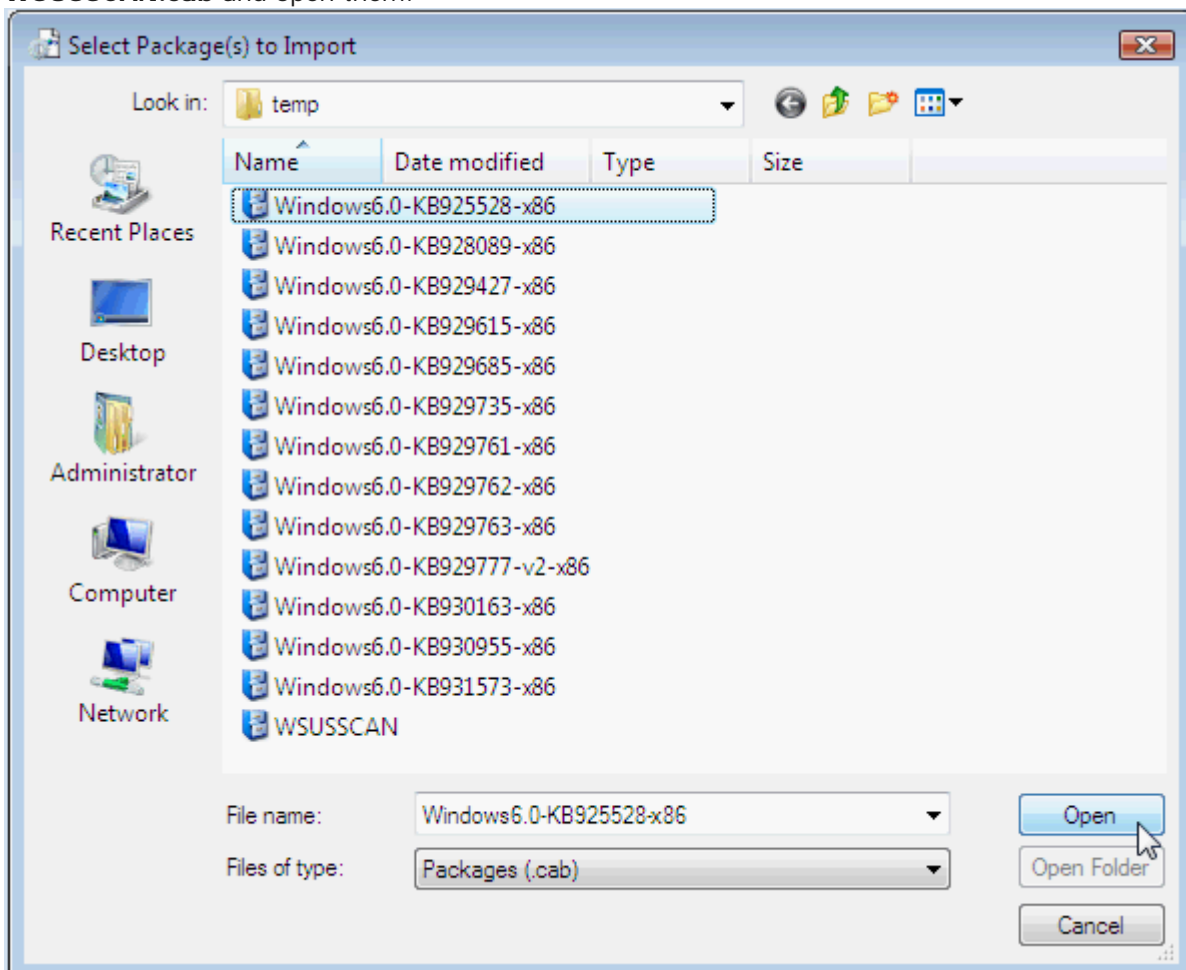
Now we **open a NEW answer file** choosing **New Answer File...** from **File menu**:



We now choose **Import Package(s)...** from **Tools menu**:



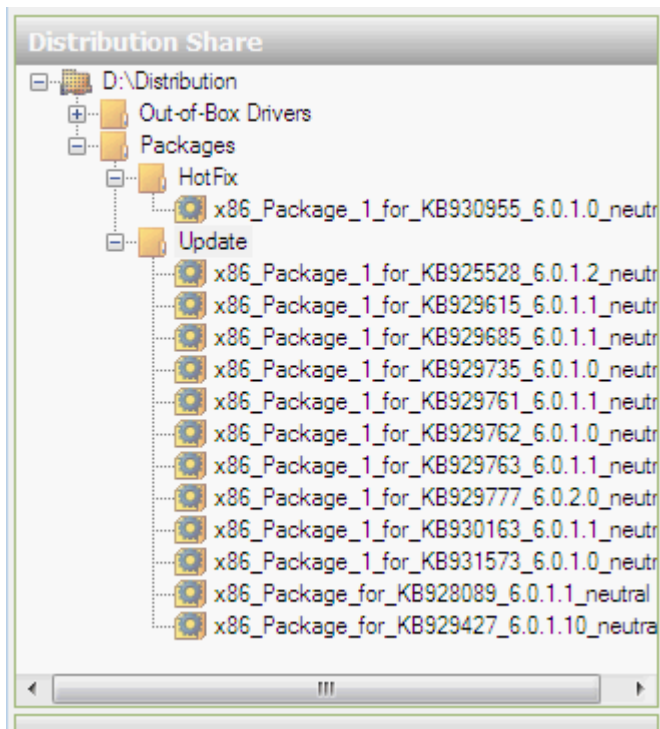
We browse to **D:\VistaWork\Updates\Temp** and **choose all present cab files except the WSUSSCAN.cab** and open them:



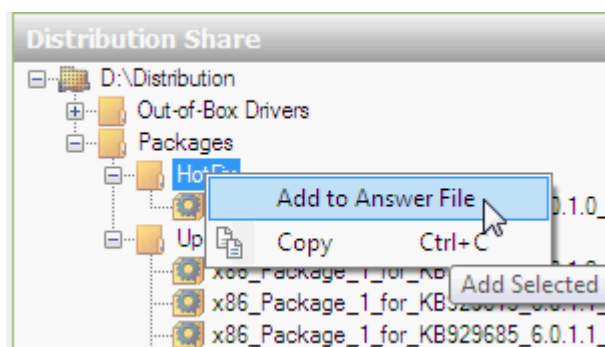
NOTE:

It seems to be, that **WSIM can't import to many cab files at the same time**. So you **may have to import them step by step** behind each other. I can import up to four packages at the same time usually.

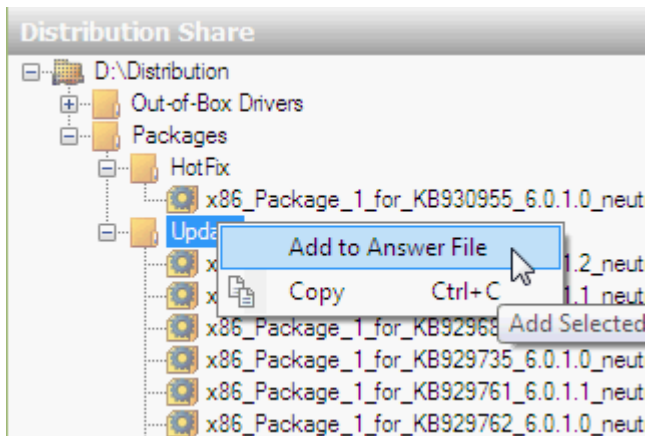
After importing all updates the Distribution Share pane will look like this:

**Step 6: Adding update and hotfix packages to answer file**

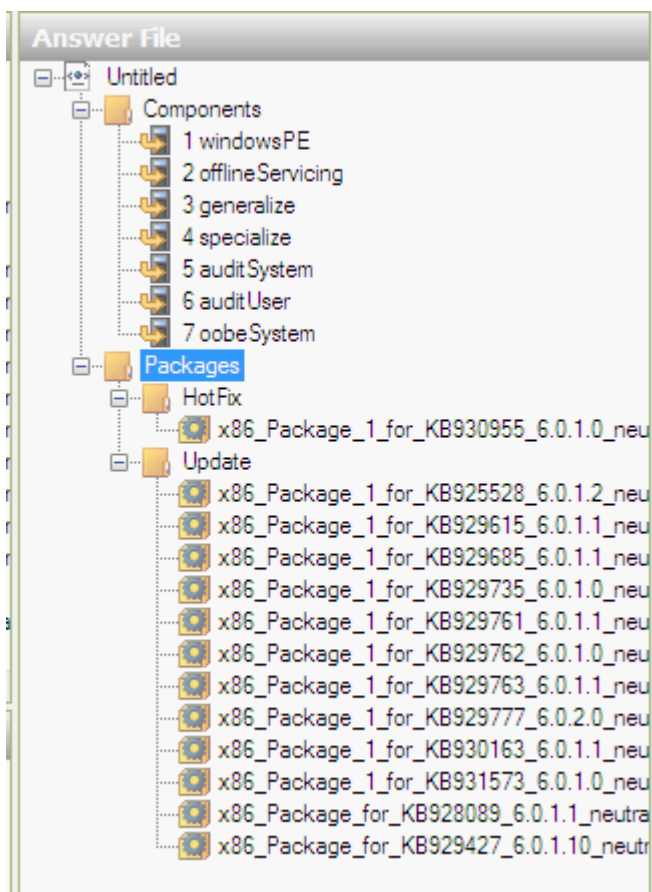
We right click on **HotFix** and choose **Add to Answer File**:



We now **do the same with Updates:**

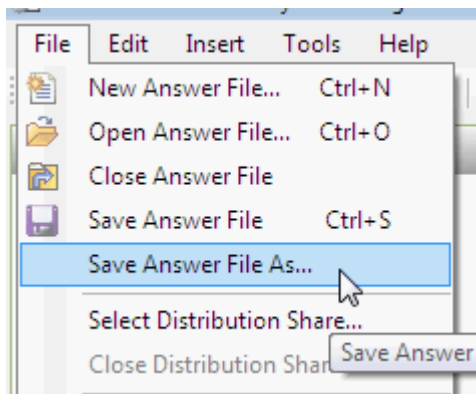


So our **Answer File** pane will look like this:

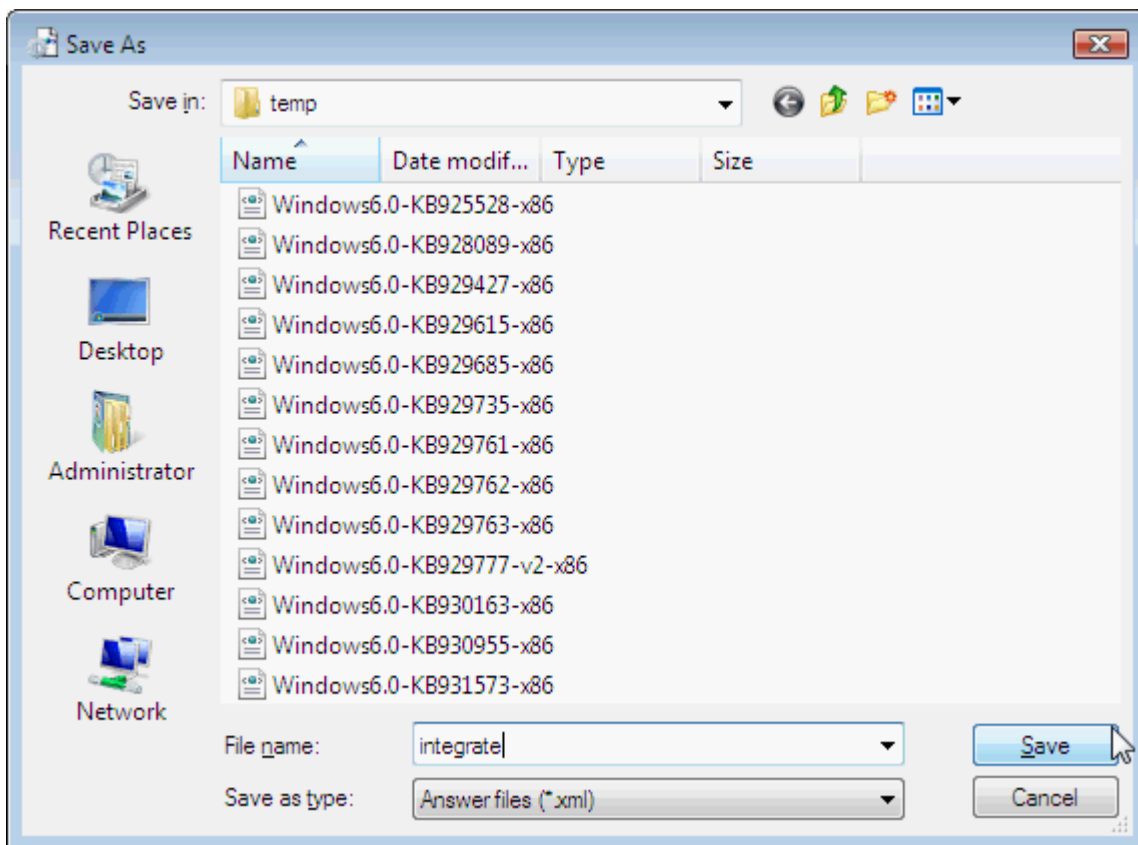


Step 7: Saving answer file as Integrate.xml

From **File** menu we choose **Save Answer File As...**,



browse to **D:\VistaWork\Updates\Temp** and save the file as **Integrate.xml**:



Step 8: Integrating updates using package manager pkgmgr.exe

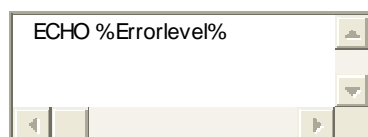
We open **Start\Program Files\Microsoft Windows AIK \Windows PE Tools Command Prompt** to insert the following command line:

```
START "PMgr" /WAIT "%Programfiles%\Windows AIK\Tools\Servicing\pkgmgr.exe"  
/o:D:\VistaWork\Mount;D:\VistaWork\Mount\Windows /n:D:\VistaWork\Updates\Temp\integrate.xml  
/s:D:\VistaWork\Sandbox /l:D:\VistaWork\Logs\integrate_updates.log
```

IMPORTANT:

This process will extract the cab files, check dependency and installs updates into Mount directory. Depending on number of updates integrated and layout of your system it will take several minutes to finish! So **DO NOT CANCEL** this procedure!!!

If command prompt is available again we type



to check, if everything went right. A "0" is displayed if everything went right.

If we receive something else than "0", something went wrong during integration. We open **Explorer** and browse to **D:\VistaWork\Logs** and open the **integrate_updates.log.txt** to get some more detailed information.

Step 9: Save changes to install.wim

Only if **Errorlevel** is "0" we will save the changes to install.wim:

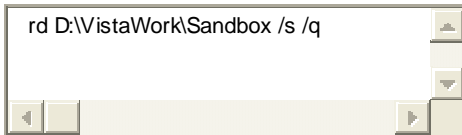
```
imagex /unmount /commit D:\VistaWork\Mount
```

IMPORTANT:

Again it will **take a while** to save the changes so **do not cancel!**

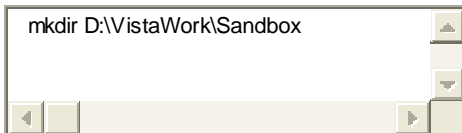
Step 10: Clearing Sandbox directory

It's **important to clear the Sandbox folder** once a **pkgmgr** process has finished, **does not matter if it was successful or not**. We type



```
rd D:\VistaWork\Sandbox /s /q
```

to **remove** the **Sandbox** folder, and



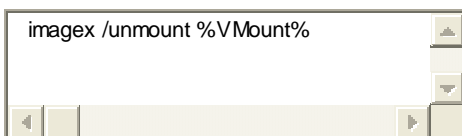
```
mkdir D:\VistaWork\Sandbox
```

to **create** it **again**.

Now we are finished!

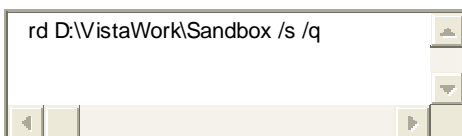
If something went wrong

If we **have not got Errorlevel "0"**, we **close** the **install.wim** **without saving changes**:



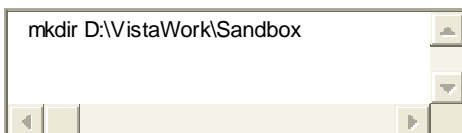
```
imagex /unmount %VMount%
```

Now we **remove** the **Sandbox** folder,



```
rd D:\VistaWork\Sandbox /s /q
```

and **recreate** it:

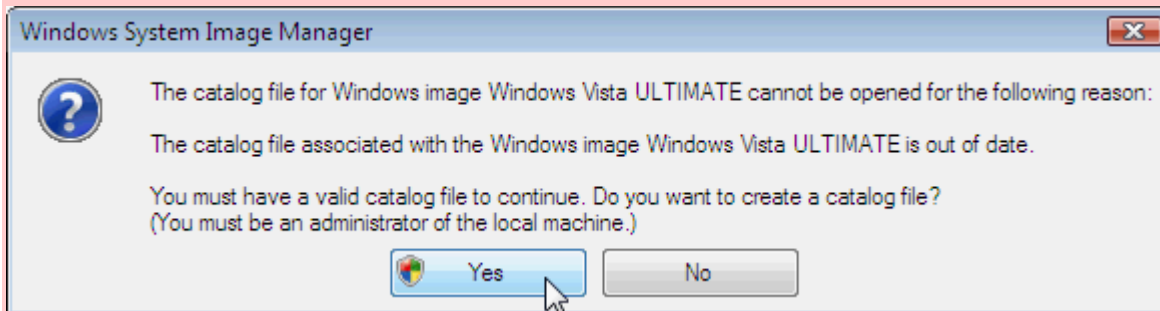


```
mkdir D:\VistaWork\Sandbox
```

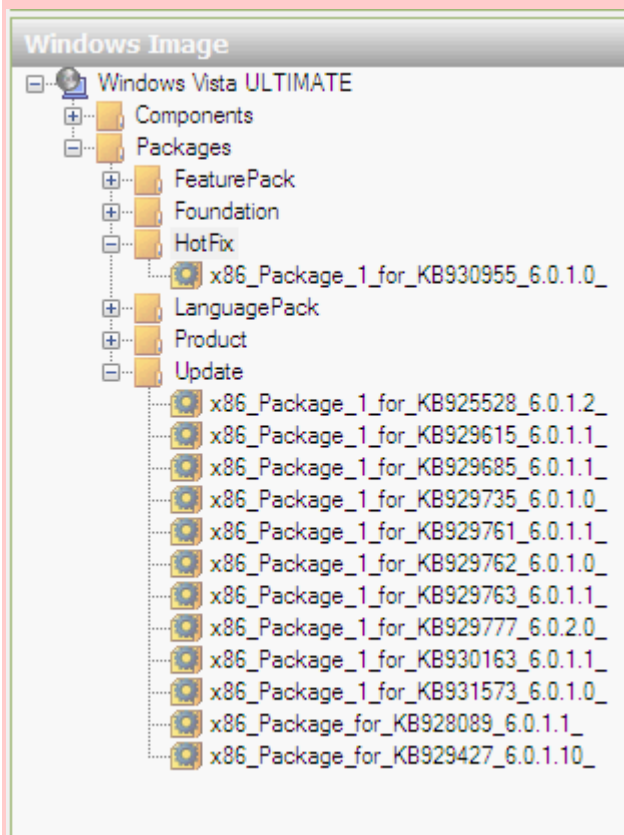
We **mount** the **install.wim** again (Step 2) and **repeat Step 8 afterwards**. If it's still failing, the update files may be corrupted and we need to download them again.

Important final note:

If we open the **changed install.wim** in **WSIM** again, we **need to rebuild the catalog file**:



Afterwards **Windows Image pane** is **presenting the update files**:



The update integration has finished successfully!

Alternative II - Integrating updates using a batch

This batch **requires**, that you have installed the **VU Batch Modules** before and that **install.wim** is **mounted** to **\\VistaWork\Mount**.

If you have the **VU Batch Modules installed** on your system you can mount **install.wim** by **executing** **mount_install_wim.cmd**.

After that you can let run the batch below:

NOTE:

This batch **needs adjustments** of the following line **depending on** the **OS language** you're using:

For /F "Tokens=5" %%i IN ('FINDSTR ".cab" Sort.txt') DO Call :WriteXML %%i

Read **comments inside offline_update.cmd** to find out more about!

offline_update.cmd:

```
@Echo off
TITLE Vista Update Integration (offline)

:-----
::Purpose: Batch integrates msu update files into install.wim offline
::
::CALL: Called by user
::
::Requirements: install.wim has to be mounted to %VMount% already!!!
::
::Version: 1.0.1
::
::History:
::
::Changes by 1.0.1:
:: 1. Added further Explanations for pkgmgr.exe call
:: 2. Added further Explanations for "DIR /OD >sort.txt" line
:: 3. Changed the value in the following line, which was
::     For /F "Tokens=4" %%i IN ('FINDSTR ".cab" Sort.txt') DO Call :WriteXML %%i
::     and is now
::     For /F "Tokens=5" %%i IN ('FINDSTR ".cab" Sort.txt') DO Call :WriteXML %%i
::     Change was necessary to get it worked with English OS
:-----

::Check environment
::Fist of all is checked, if all used environment variables are declared and valid and if
::all used directories do exist. This done by check_umgebung.cmd.
::
::check_umgebung.cmd will assign a value "true" to EnvErr variable, if there is an error in
::working environment and it will assign "false" to EnvErr variable, if everything is alright
::in working evrionment.
::
::If there is an error in working environment (EnvErr=true) au_fehler.cmd is called.
::au_fehler.cmd will display an error message and quit the batch.

CALL check_umgebung.cmd
IF %EnvErr%==true CALL au_fehler.cmd

::You need to adpat the following SET lines, if your system was not prepared or not prepared
::correctly using preplab.cmd. Remove everything in front of SET and adjust the path behind
::"=". Expect PMgrDir - this is path where pkgmgr.exe is located - all pahtes are arbitrary.
:----- Start Adjustment -----
::SET PMgrDir=%Programfiles%\Windows AIK\Tools\Serviceing
::SET VMount=C:\WinPERE\Mount
::SET VUpdates=J:\Vista\Updates
::SET VSand=C:\WinPERE\Sandbox
::SET VLog=C:\WinPERE
:----- End Adjustment -----

::The nex step is necessary, cause Expand.exe is inside different folders depending on OS.
::In Vista Expand.exe is located in %Windir%\System32 in XP it's located inside same
```

```

::folder like package manager (%PMGrDir%).

SET Extract=%PMGrDir%
IF NOT EXIST "%PMGrDir%\Expand.exe SET Extract=%Windir%\System32

::
:: _____
::      MAIN PROCEDURE
:: _____

::Creating an temporary directory inside Updates folder. The msu files will be extracted
::to that new temporary folder.

IF NOT EXIST %VUpdates%\Temp (MKDIR %VUpdates%\Temp)
DEL %VUpdates%\Temp\*. * /q

::PUSHD will jump to %VUpdates% folder. Than all present files inside Updates folder are
::assigned to :Extract subroutine.

PUSHD %VUpdates%
FOR %%i IN (*) DO (Call :Extract %%i)

::Checking for older file version created by this batch before may be and deleting them
::if necessary to avoid any conflicts.

IF EXIST %VUpdates%\Temp\integrate.xml DEL /Q %VUpdates%\Temp\integrate.xml
IF EXIST %VUpdates%\Temp\Sort.txt DEL /Q %VUpdates%\Temp\Sort.txt

::The following three ECHO lines are redirected into integrate.xml file.
::So they are building the first lines of integrate.xml.

ECHO ^<^?xml version="1.0" encoding="utf-8"?^>^ >>%VUpdates%\Temp\integrate.xml
ECHO ^<^unattend xmlns="urn:schemas-microsoft-com:unattend"^>^ >>%VUpdates%\Temp\integrate.xml
ECHO      ^<^servicing^>^ >>%VUpdates%\Temp\integrate.xml

::PUSHD jumps into %VUpdates%\Temp folder.
::Dir /OD lists all the files inside Temp folder by date. The output is redirected to Sort.txt.
::
::NOTE:
::Dir /OD output can be different depending on OS language you're using, so you may need to adjust
::the Tokens value in the following line!
::
::Example:
::Dir /OD output on German OS:
::
::12.02.2007  00:25          4.321.981 Windows6.0-KB929427-x86.msu
::12.02.2007  00:27          394.535 Windows6.0-KB925528-x86.msu
::      |          |          |          |
::Token 1  Token 2          Token 3          Token 4
::
::
::Dir /OD output on English OS:
::
::
::              Token 3
::              |
::12.02.2007  00:25 AM          4.321.981 Windows6.0-KB929427-x86.msu
::12.02.2007  00:27 AM          394.535 Windows6.0-KB925528-x86.msu
::      |          |          |          |
::Token 1  Token 2          Token 4          Token 5
::
::
::The following FOR loop is searching cab files inside the sort.txt list and forwards the
::file name of them (sorted out by Tokens value) - if found - to WriteXML subroutine.

PUSHD %VUpdates%\Temp
Dir /OD > Sort.txt
For /F "Tokens=5" %%i IN ('FINDSTR ".cab" Sort.txt') DO Call :WriteXML %%i

::Now the two finishing lines are added to integrate.xml. Again the ECHO commands are
::redirected to do so.

ECHO      ^<^/servicing^>^ >>%VUpdates%\Temp\integrate.xml
ECHO ^<^/unattend^>^ >>%VUpdates%\Temp\integrate.xml

```

```

::At least package manager (pkgmgr.exe) is executed and installs the packages using following
::parameters:
::
::/o: points to the folder where image is mounted to and defines the windows folder inside
:: the mounted image.
::
::/n: assigns the answer file (xml file) containing the informations about the updates, which
:: supposed to be installed.
::
::/s: points to the sandbox folder (temporary folder) which will be used by package manager
:: to extract the cabs before integrating them.
::
::/l: points to the location of the log file and defines the name of the log.
::
::IMPORTANT:
::START command does interpret the first Expression inside quotes as Window Title. This is
::the reason, why you need "PMgr". You can rename but not remove it. Otherwise the whole
::command will fail!!!

START "PMgr" /WAIT "%PMgrDir%\pkgmgr.exe" /o:%VMount%;%VMount%\Windows
/n:%VUpdates%\Temp\integrate.xml /s:%VSandb% /l:%VLog%\pkglog.txt
ECHO The process of integration was finished with error level %ERRORLEVEL%.
RD %VSandb% /q /s
DEL %VUpdates%\Temp\*. * /q
MKDIR %VWork%\Sandbox
Pause
EXIT

::
:: _____
:: SUB PROCEDURES & PARTS EXECUTED OPTIONAL
:: _____

::Extract subroutine extracts the msu files die MSU-Dateien from %VUpdates% directory
::to %VUpdates%\Temp. There they will be present with same names but cab format.

:Extract
START "Expand" /WAIT "%Extract%\Expand.exe %1 -f:* %VUpdates%\Temp
GOTO :EOF

::WriteXML subroutine writes entries into integrate.xml, which are specified by the name of
::corresponding cab file. This is the command line, which will call the installation of the
::cab during pkgmgr procedure later.

:WriteXML
IF "%1"=="WSUSSCAN.cab" Goto :EOF
SET Name=%1
IF "%Name%"=="." GOTO :EOF
SET XML=%Name:~0,-4%.xml
SET Cab=%Name:~0,-4%.cab
ECHO ^<^package action="install"^>^ >%VUpdates%\Temp\integrate.xml
FOR /F "Tokens=*" %%i IN ('FINDSTR "assemblyIdentity" %XML%') DO (Echo %%i
>%VUpdates%\Temp\integrate.xml)
ECHO ^<^source location="%VUpdates%\Temp\Cab%" />^ >%VUpdates%\Temp\integrate.xml
ECHO ^<^/package^>^ >%VUpdates%\Temp\integrate.xml
:EOF

```

NOTE:

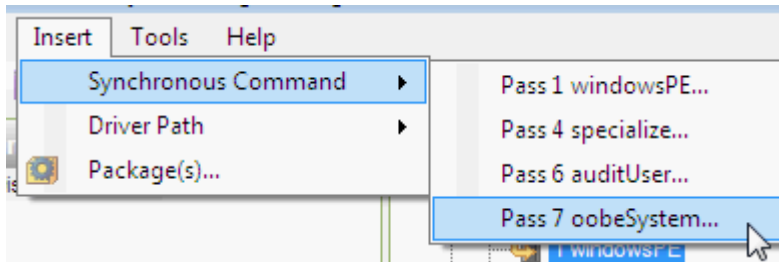
To save changes you need to let run unmount_commit.cmd!

HowTo: Installing applications during unattended Vista setup

Introduction

In the following we will **adapt** our **Autounattend.xml** and **removable media**, so we will be able to **run application installations from removable media directly**.

The installation and execution of applications and / or scripts during Vista setup is unified now. Everything will be executed using a **Synchronous Command**. We can **add** a such a command form **Insert menu** in **WSIM**:



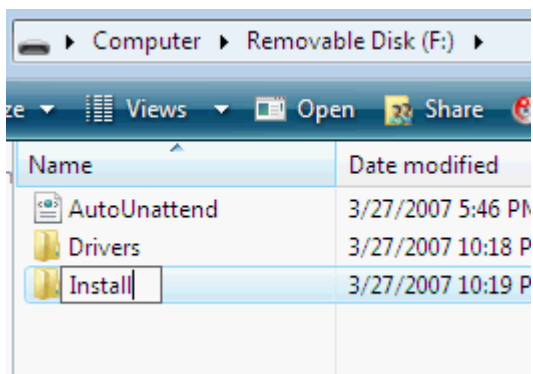
Synchronous Commands are **processed sequential**. The **next command** is always **started only**, **after** the **previous** was **finished**. Synchronous Commands are **running in user context**. They are **executed right after logon** but **before Desktop is loaded** finally, and they are **executed once only**.

What we need

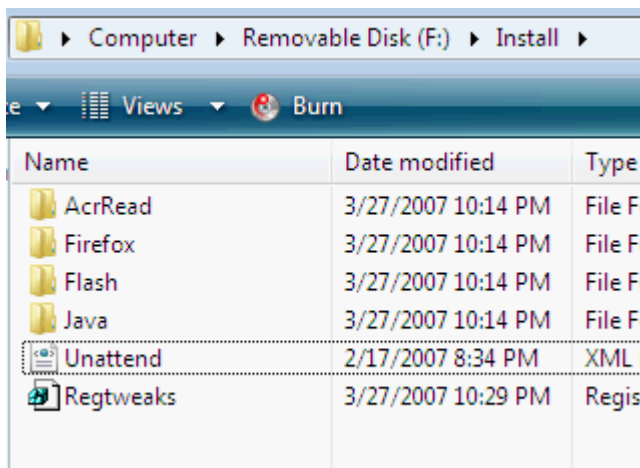
1. **WAIK** installed
2. Basic **Autounattend.xml**
3. **Additional Applications** (eg. Adobe Reader, Firefox etc.)

Step 1: Creating an Install folder on removable media

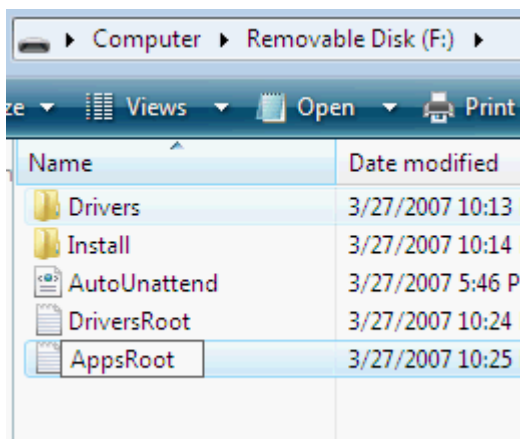
First of all we need to **create a new folder** in the **root of** our **removable media**, which will hold the installer packages of the applications we want to install.



We **open** the **Install folder** and **create** a **subfolder for every application** we want to install. We **copy** the **installation files** of the corresponding application **to these folders** than:



We **navigate back** to the **root of removable media** and **create** a **new Text Document** called **AppsRoot.txt**



EXPLANATION:

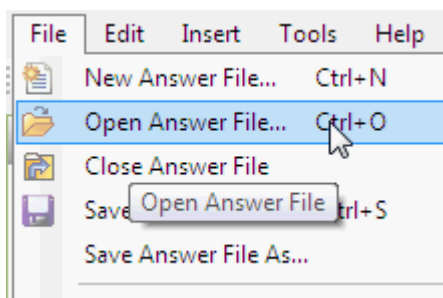
The **AppsRoot.txt** is used to identify our **removable media** during Vista setup. So we can **find out and point to the Install folder**.

NOTE:

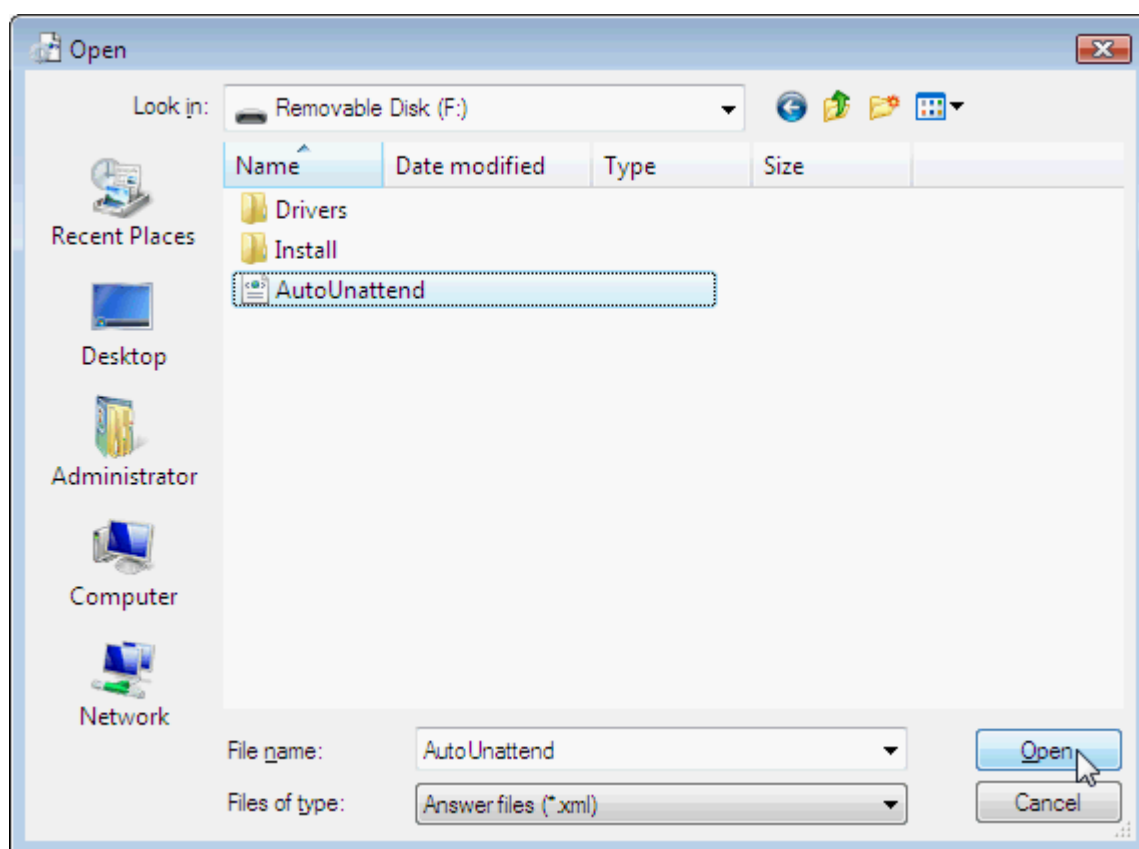
You don't need to store the Install folder to a separate removable media. You can put it - including the AppsRoot.txt - to the Vista DVD, too. But it's more convenient to have all these separate files on a USB flash drive in my opinion.

Step 2: Editing answer file (Autounattend.xml)

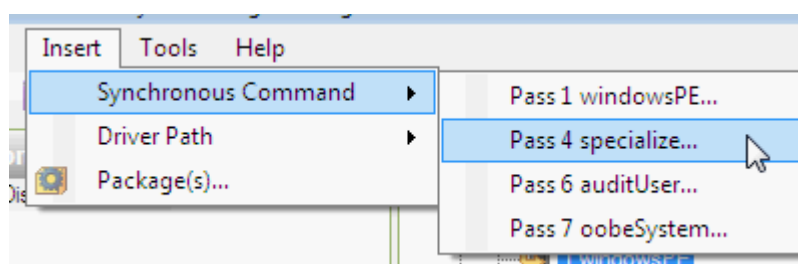
We **open WSIM**, choose **Open Answer File...** from **File** menu,



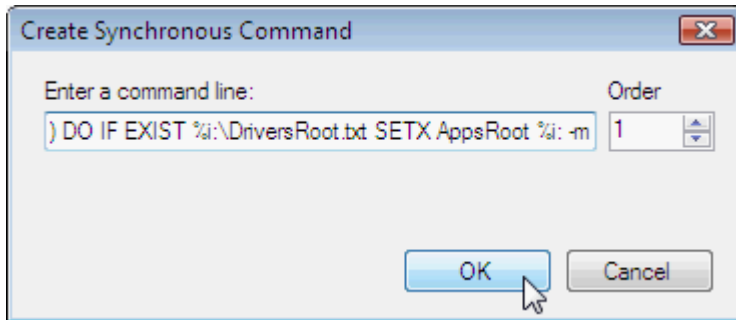
and **browse** to our **Autounattend.xml** on **removable media**:



First of all we **choose Synchronous Command \ Pass 4 specialize** from **Insert** menu:



We type the **following command** into the opening **dialog** window:

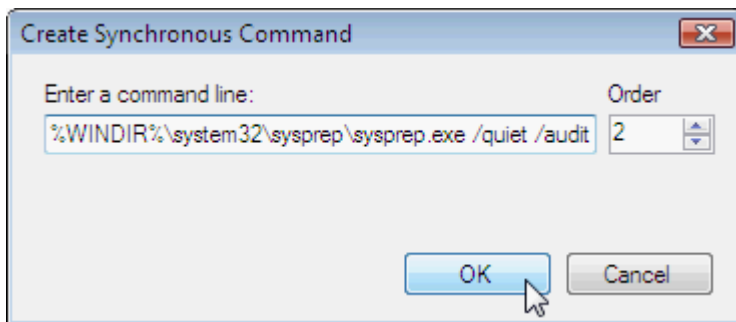


```
cmd /c "FOR %i IN ( C D E F G H I J K L N M O P Q R S T U V W X Y Z ) DO IF EXIST %i:\AppsRoot.txt  
SETX AppsRoot %i: -m"
```

EXPLANATION:

This command will set an **environment variable** %AppsRoot% to the system environment. %AppsRoot% variable **points to the drive where the AppsRoot.txt is located**. So we can use it to point to our removable media during installation.

We insert a **second Synchronous Command** to **Pass 4 specialize**:

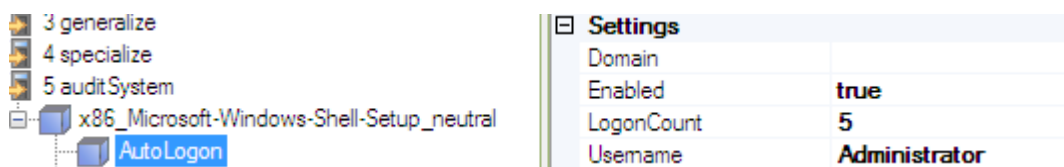


```
%WINDIR%\system32\sysprep\sysprep.exe /quiet /audit
```

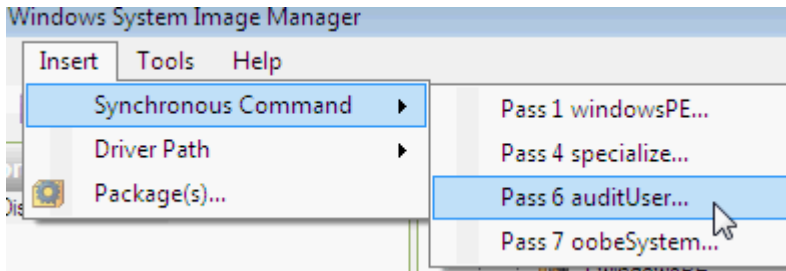
EXPLANATION:

This command will **boot system to audit modes**, where the applications will be installed.

Now we **create** an **AutoLogon** for the Administrator account in audit mode. So we add **Windows-Shell-Setup \ AutoLogon** component to **auditSystem** pass:



Next we **choose Synchronous Command \ Pass 6 auditUser** from **Insert menu**:



NOTE:

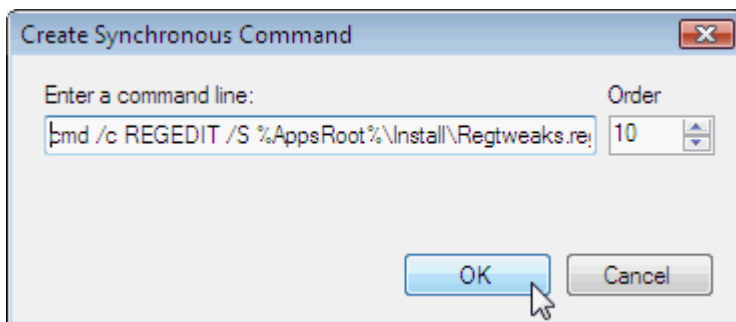
We **need to repeat** this step for **every** further **application** we want to install or for **every script** we want to let run **during setup!**

The opening dialog asks for the command to run. This is the application installer call including silent and other switches. With Order property we can adjust the order applications will be installed or scripts will be run later during setup.

Examples

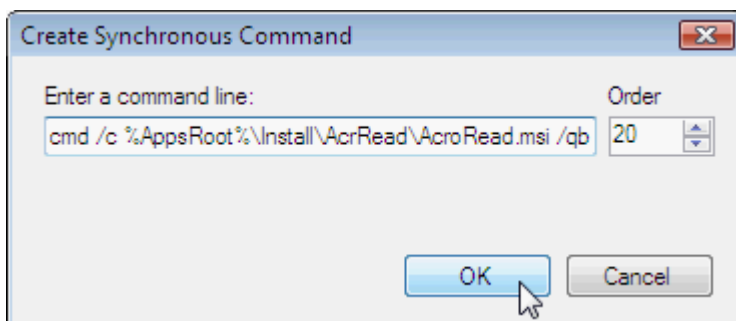
The following **examples** are showing the settings for **importing Regtweaks.reg** and **setup calls** of **Adobe Reader** (extracted already), **Mozilla Firefox**, **Adobe Flash Player** and **Sun Java VM**:

Regtweaks.reg:



```
cmd /c REGEDIT /S %AppsRoot%\Install\Regtweaks.reg
```

Adobe Reader (extracted already):

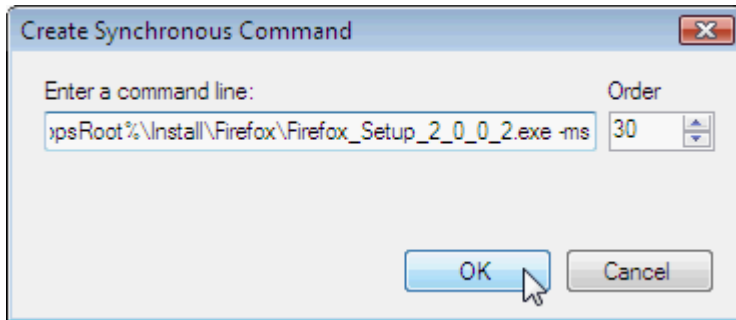


```
cmd /c %AppsRoot%\Install\AcrRead\AcroRead.msi /qb
```

NOTE:

Even if I was using the main administrator account (Administrator) during setup, I've had problems to install the Reader, when using the downloaded installer. It seems to be, that there is a problem accessing Temp folder during unattended setup. After extracting the Reader installer to Install folder - so I had the .msi file - it was working.

Mozilla Firefox:

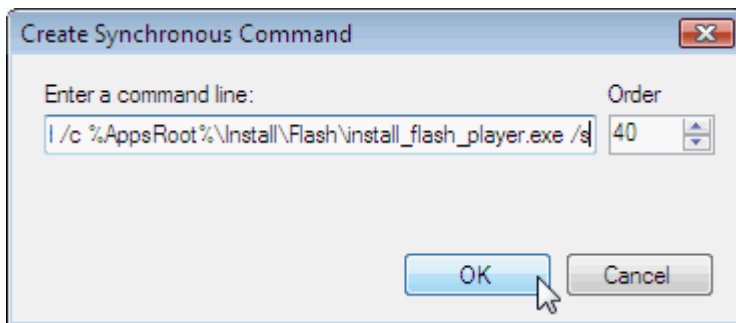


```
cmd /c %AppsRoot%\Install\Firefox\Firefox_Setup_2_0_0_2.exe -ms
```

NOTE:

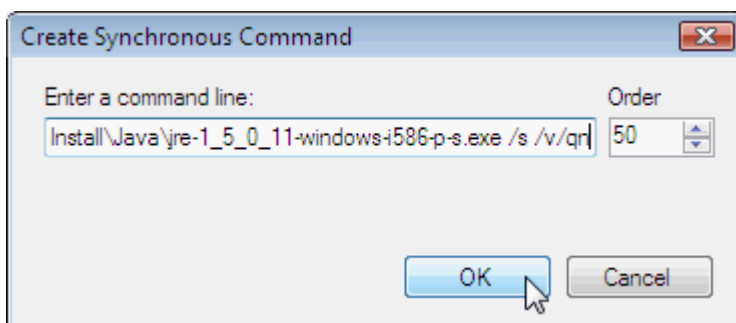
I've replaced the "." through "_" inside the name of Firefox setup file!

Adobe Flashplayer:



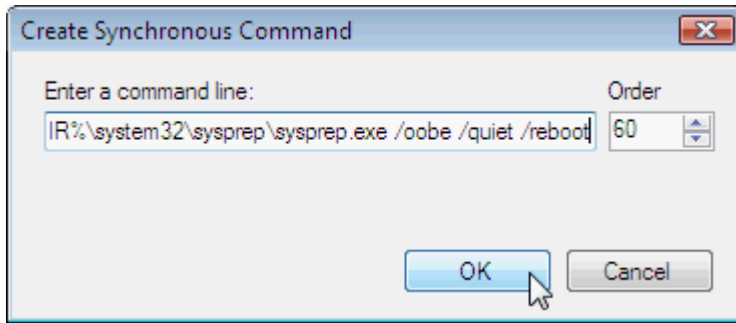
```
cmd /c %AppsRoot%\Install\Flash\install_flash_player.exe /s
```

Sun Jave VM:



```
cmd /c %AppsRoot%\Install\Java\jre-1_5_0_11-windows-i586-p-s.exe /s /v/qn
```

Leave audit modes and boot to oobe:

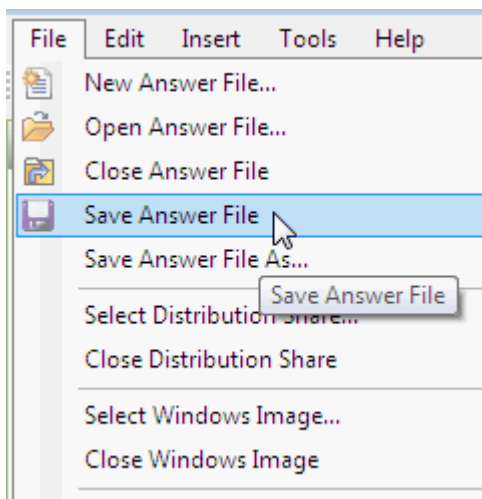


```
%WINDIR%\system32\sysprep\sysprep.exe /quiet /oobe /reboot
```

NOTE:

This last command has always to be set **on last position**, cause it will **leave the audit modes** and will **boot to oobe**.

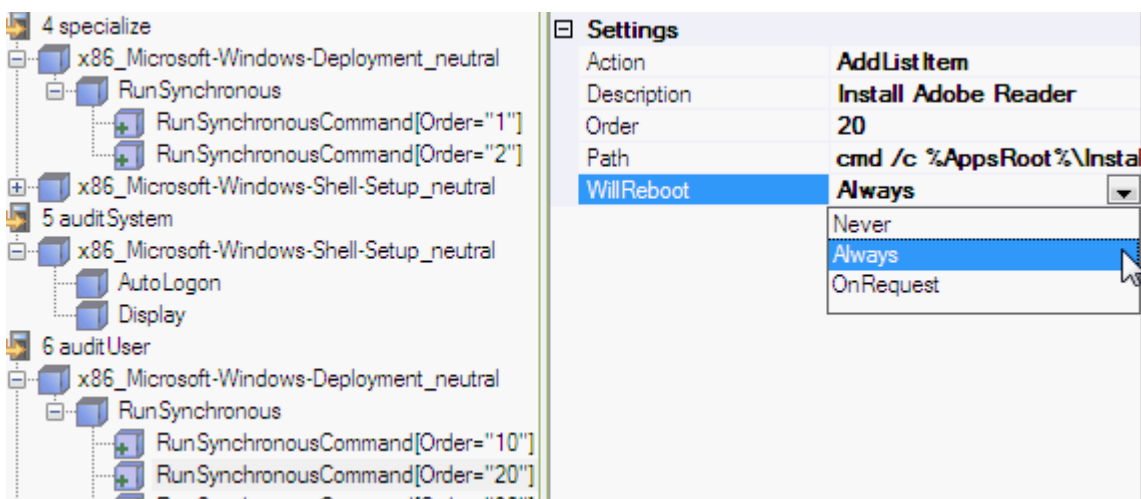
Now we **save** our **Autounattend.xml**,



and we are **ready to roll out** our unattended setup with applications.

How to do a reboot between application installs?

If you need to do a reboot between installation of different applications **you can set WillReboot to Always in Properties pane:**



HowTo: Importing Registry Tweaks during Vista setup

Introduction

Regtweaks in Vista are - same manner **like 2000/XP** - imported into registry using a **.reg file** (below called **Regtweaks.reg**). The structure of Vista **Regtweaks.reg** does not differ from structure of **2000/XP Regtweaks.reg**.

A lot of Regtweaks for 2000/XP will work for Vista, too - but not all of them unfortunately.

What we need

1. **WAIK** installed
2. A basic and properly working **Autounattend.xml**

The structure of Regtweaks.reg

```
Windows Registry Editor Version 5.00

;Ausblenden des Wortes 'Verknüpfung' bei Verknüpfungen
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer]
"link"=hex:00,00,00,00

;'In Ordner kopieren' zum Rechtsklick-Menü hinzufügen
[HKEY_CLASSES_ROOT\AllFilesystemObjects\shellex\ContextMenuHandlers\{C2FBB630-2971-11D1-A18C-00C04FD75D13}]

;'In Ordner verschieben' zum Rechtsklick-Menü hinzufügen
[HKEY_CLASSES_ROOT\AllFilesystemObjects\shellex\ContextMenuHandlers\{C2FBB631-2971-11D1-A18C-00C04FD75D13}]

;Browsen mit dem IE beschleunigen
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\AFD\Parameters]
"BufferMultiplier"=dword:00000400
```

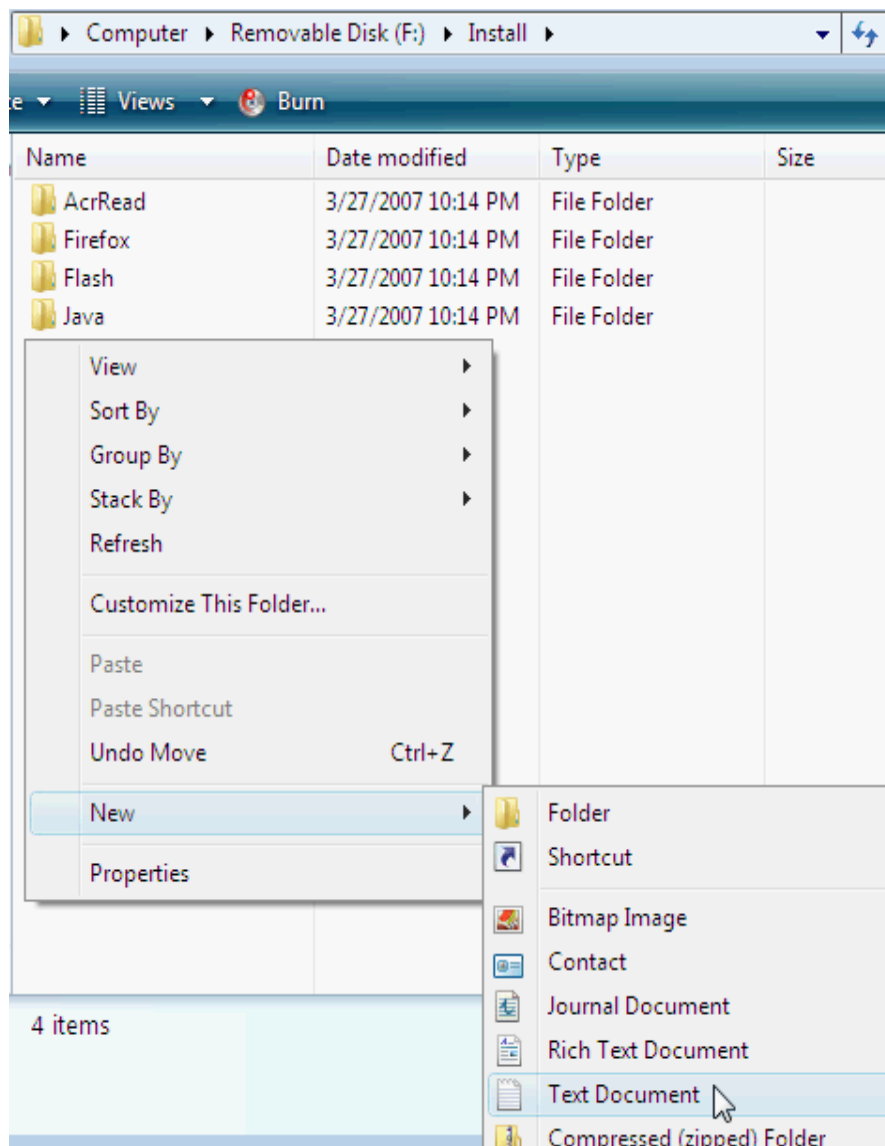
IMPORTANT:

The file **must** always start with the same line - **Windows Registry Editor Version 5.00**.

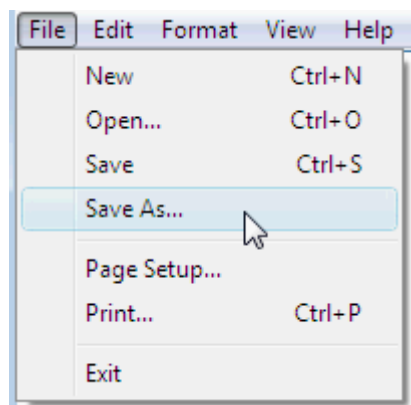
You can add comments behind the ";".

Step 1: Creating and saving a Regtweaks.reg

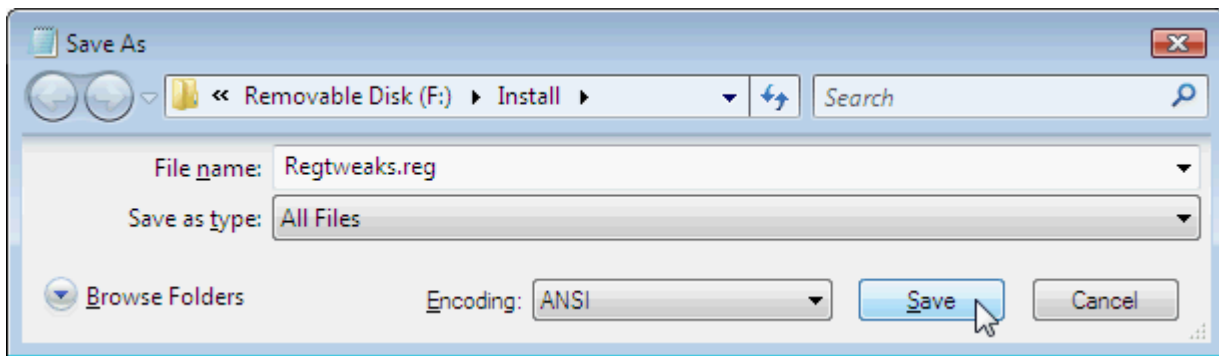
We open **Windows Explorer** and **browse to** the **Install folder** on our removable media. If you don't have a **Install** folder yet, create one in the root of removable media. We open **Install folder** and **right click** on **empty space**. From **context menu** we choose **New \ Text Document**:



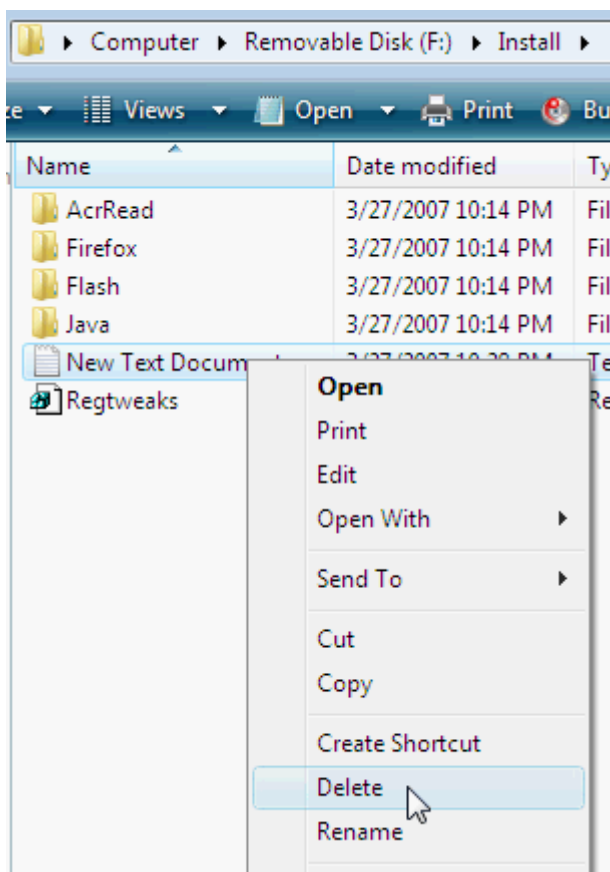
We **double-click** this new text file **to open** it. We copy the **lines** from **above** to the **file**. If you need more or other tweaks just [check out here](#). When we're finished editing the file, we choose **Save as...** from **File** menu.



We ensure that **All Files** is selected as **Save As Filetype** and we insert **Regtweaks.reg** as **Filename**.
Than we **click** on **Save**.

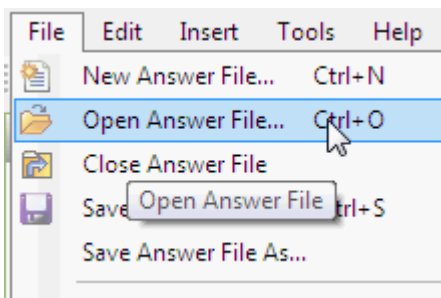


At least we will **delete** the **New Textdocument** we have created before by **right clicking** it:

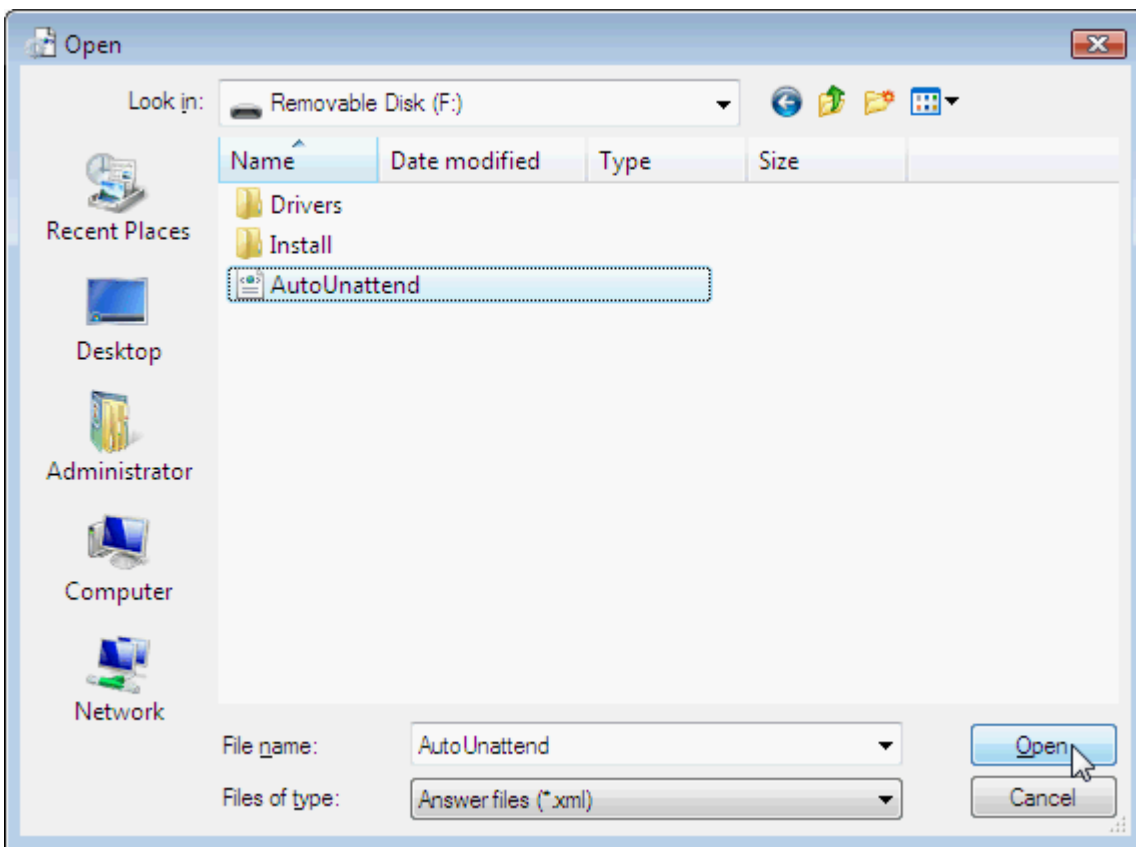


Step 2: Preparing the registry import of Regtweaks.reg using WSIM

We **start** the **WSIM** and choose **Open Answer File...** from **File** menu:



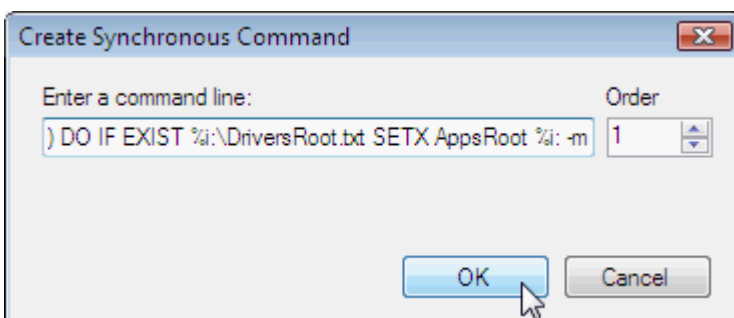
we browse to the root of our removable media, where **Autounattend.xml** is located:



NOTE:

If your Autounattend.xml is located on Vista DVD, browse to the folder where your Vista DVD is copied to.

We type the **following command** into the opening **dialog** window:

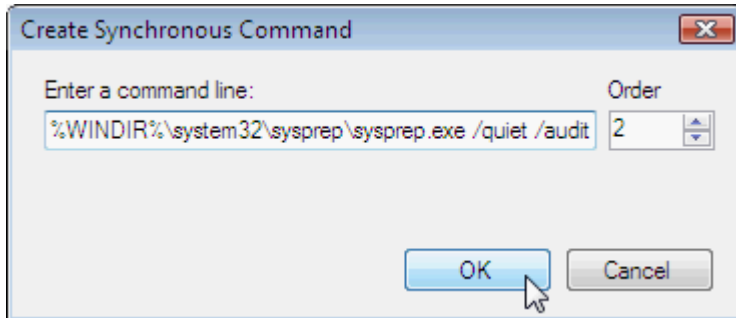


```
cmd /c "FOR %i IN (C D E F G H I J K L N M O P Q R S T U V W X Y Z) DO IF EXIST %i:\AppsRoot.txt  
SETX AppsRoot %i: -m"
```

EXPLANATION:

This command will set an **environment variable** %AppsRoot% to the system environment. %AppsRoot% variable **points to the drive where the AppsRoot.txt is located**. So we can use it to point to our removable media during installation.

We insert a **second Synchronous Command** to **Pass 4 specialize**:

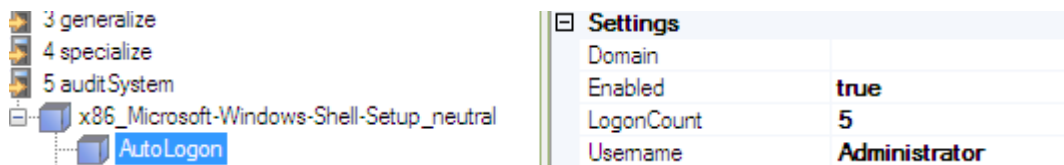


```
%WINDIR%\system32\sysprep\sysprep.exe /quiet /audit
```

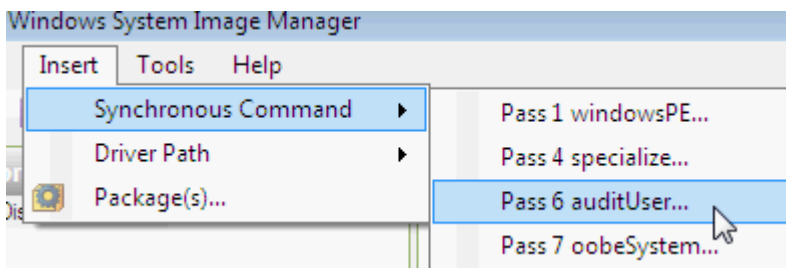
EXPLANATION:

This command will **boot** system to **audit modes**, where the applications will be installed.

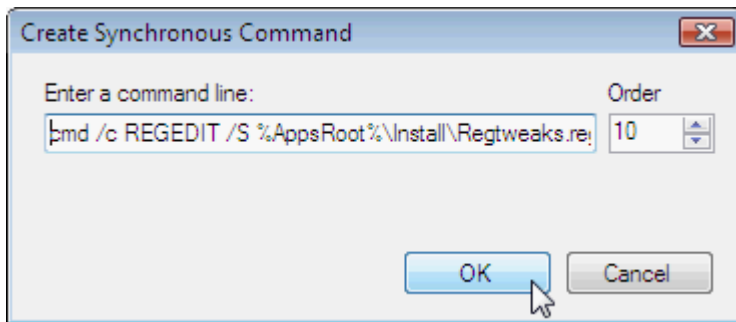
Now we **create** an **AutoLogon** for the Administrator account in audit mode. So we add **Windows-Shell-Setup \ AutoLogon** component to **auditSystem** pass:



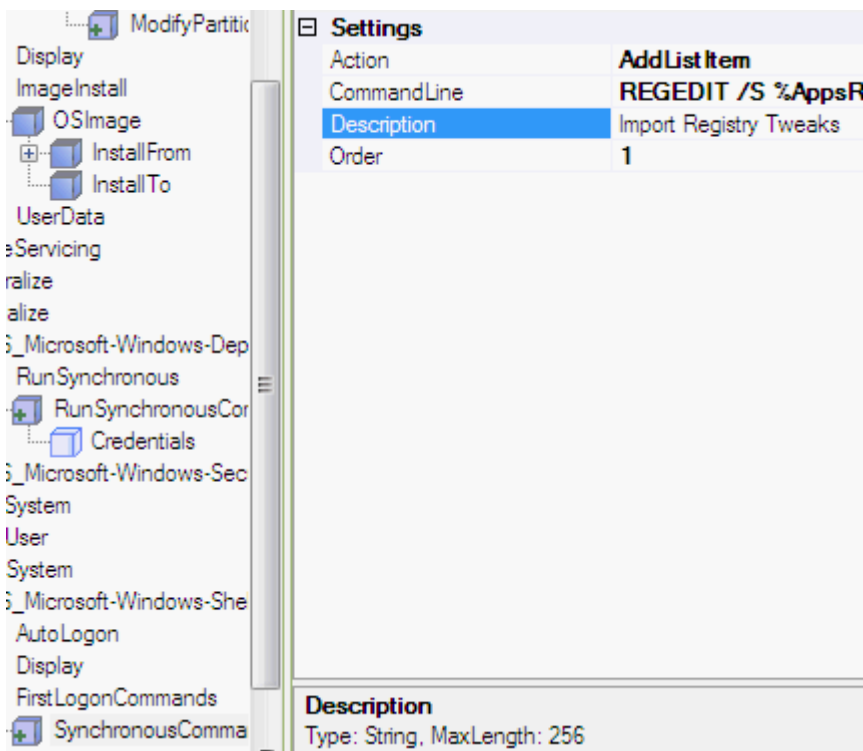
Next we choose **Synchronous Command \ Pass 6 auditUser** from **Insert** menu.



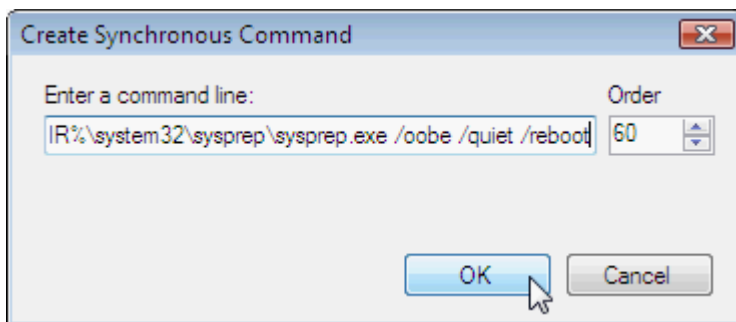
We type the **following command** into the appearing dialog box:



This will add an entry called **Windows-Deployment \ RunSynchronous** to the oobeSystem pass in **Answer File pane**. You can **add a Description** if you want, but you don't have to:



We insert a **further synchronous command** to Autounattend.xml:

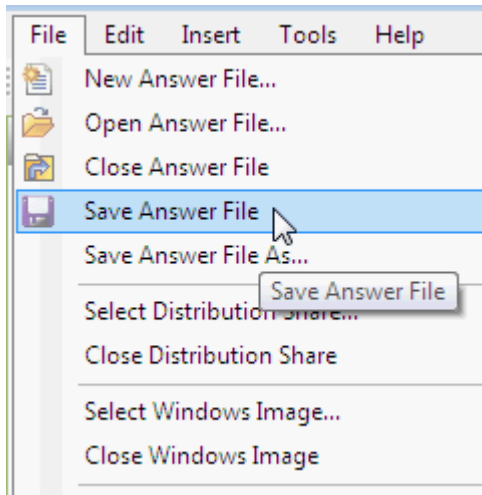


```
%WINDIR%\system32\sysprep\sysprep.exe /quiet /oobe /reboot
```

NOTE:

This last command has always to be set **on last position**, cause it will **leave the audit modes** and will **boot to oobe**.

From WSIM **File** menu choose **Save** to apply the changes to your **Autounattend.xml**.



Now everything is settled up and you can use your removable media to import the Regtweaks.reg for the logged on user profile during unattended Vista setup. The Regtweaks.reg will be imported during oobeSystem pass.

FINAL NOTE:

It's **not a must** to **import** Regtweaks.reg **during auditUser** mode.

System wide regtweaks could be applied during **specialize** pass or **oobe** pass, **too**.

User regtweaks could be applied during **oobe pass alternatively** only. During specialize pass no user account is logged on.

We ensure to **use** an **administrator account always, to import** the registry tweaks. **Otherwise** they **could be virtualized only**. That means they would be present for the session only and are not written to the registry in worst case!

How to apply registry tweaks for all further user profiles in future (Copying to Default User)?

If you want to have the registry tweaks present for all users on your system and for all users which a may created in future on your system, you need to copy the profile where you've imported the regtweaks to, to the Default User profile of Vista.

To do that you need to follow the instructions described in [Copying user profile to Default user](#) in addition to the steps we just did here.

HowTo: Copying the actual user profile over to the profile of default user

Introduction

Vista setup does allow to **copy** settings of **actual user** profile over **to** the **default user** profile (Default User). This is needed for example, if we want use **imported registry tweaks** for all users.

Copying is possible only doing **generalization using sysprep.exe**.

NOTE:

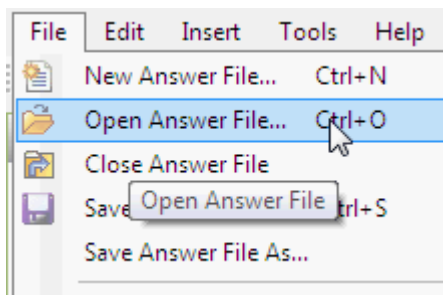
This procedure is **NOT** for **copying user profile** from **former OS installations** to Vista!

What we need

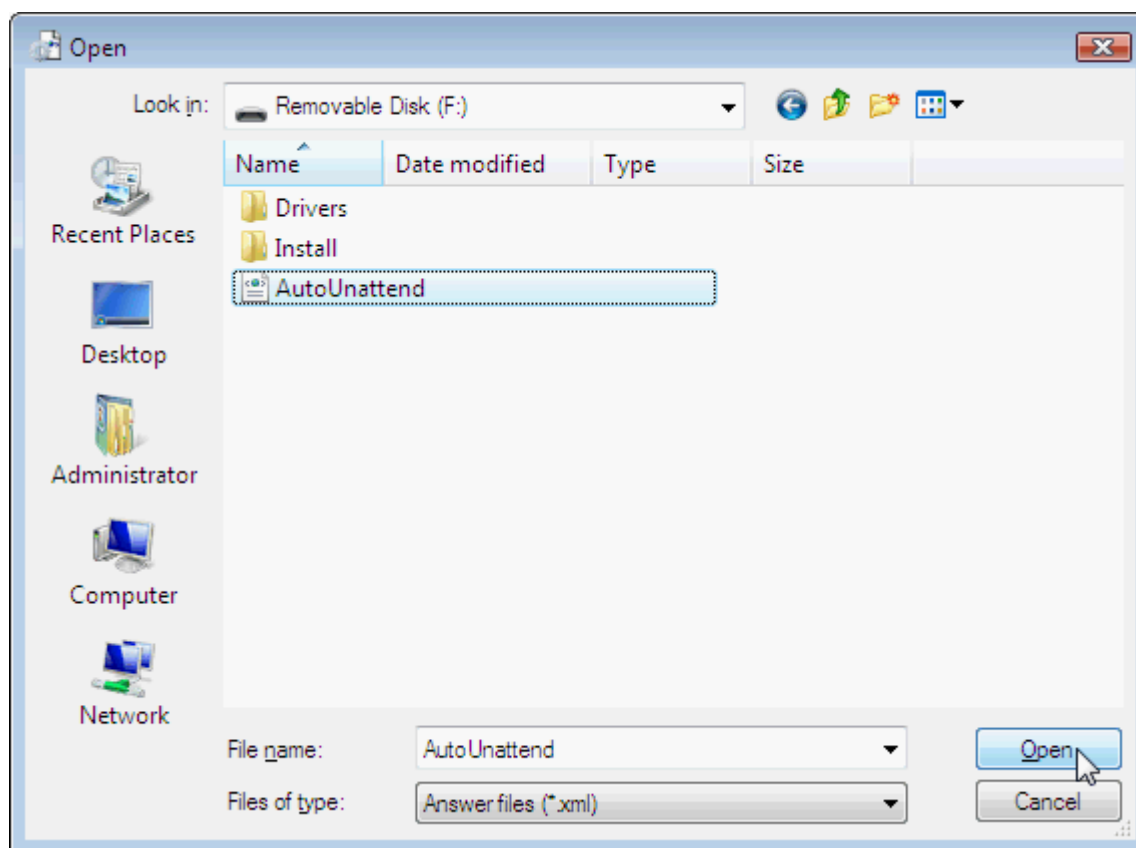
1. **WAIK** installed
2. Basic **Autounattend.xml**
3. **Imported registry tweaks**

Step 1: Executing sysprep.exe automatically for generalization

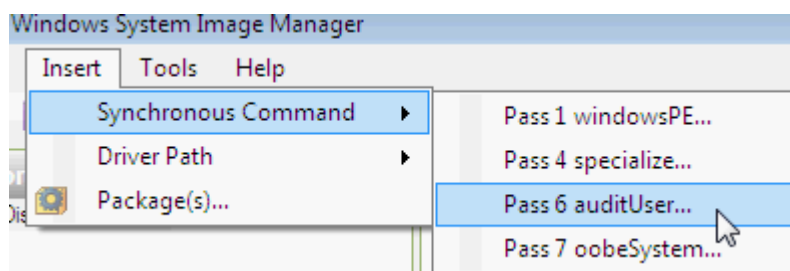
We **open WSIM** and choose **Open Answer File...** form **File** menu,



and **browse** to **Autounattend.xml** on removable media:



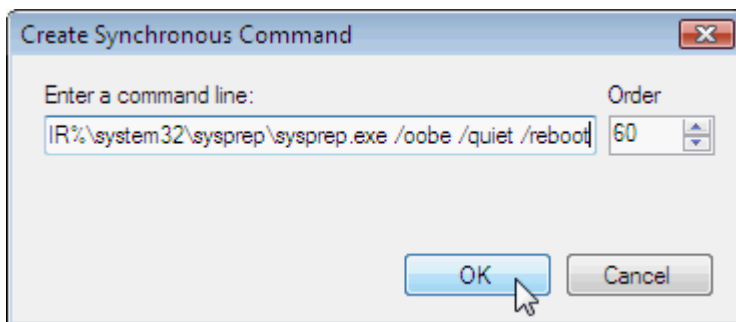
Now we **choose Synchronous Command Pass 6 auditUser** from **Insert** menu:



in **opening dialog** box we **type**:

```
%WINDIR%\system32\sysprep\sysprep.exe /generalize /quiet /oobe /reboot
```

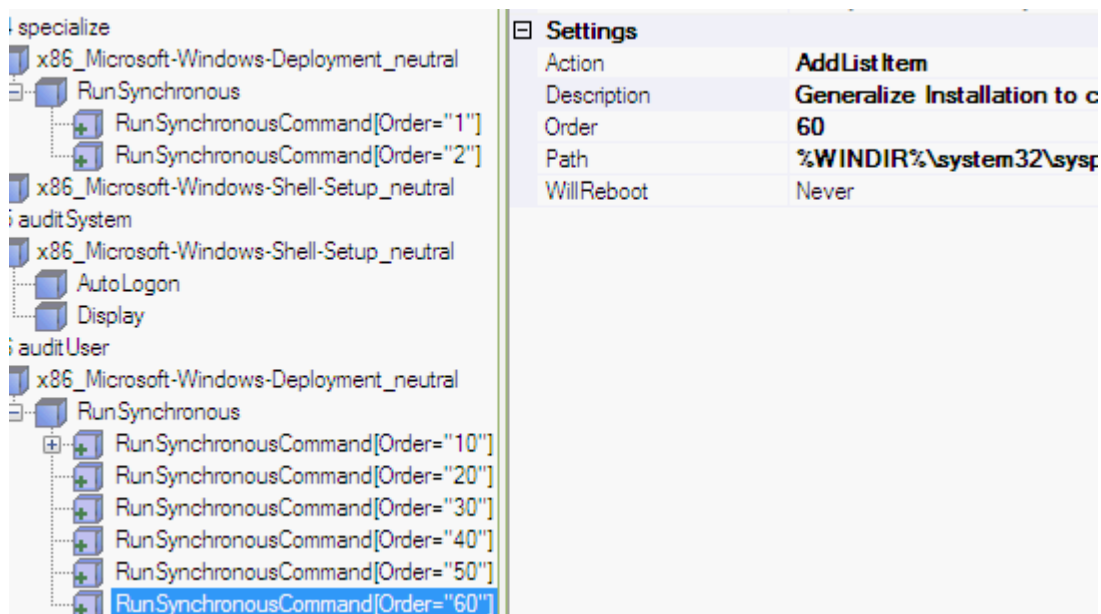
We **ensure** to set **Order** to the **last position** of our **command row**:



EXAMPLE:

If you're installing 5 applications during setup in oobeSystem pass set Order for sysprep.exe to 6. So it will be executed after everything else was installed. You can leave space between the different Order ids (eg.: 10, 20, 30, 40).

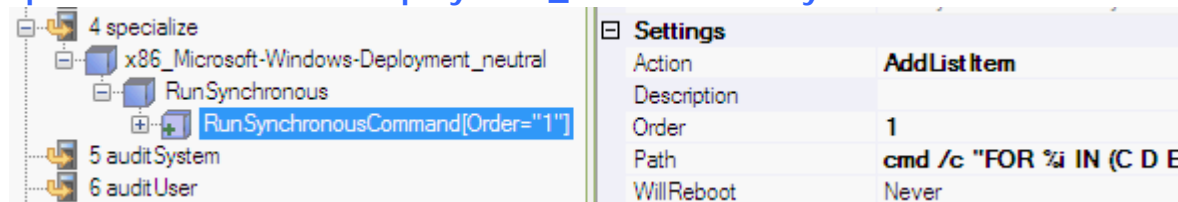
Add a **Description** if you want to:



Step 2: Checking over the settings in Autounattend.xml

Now we check over the following settings in our Autounattend.xml:

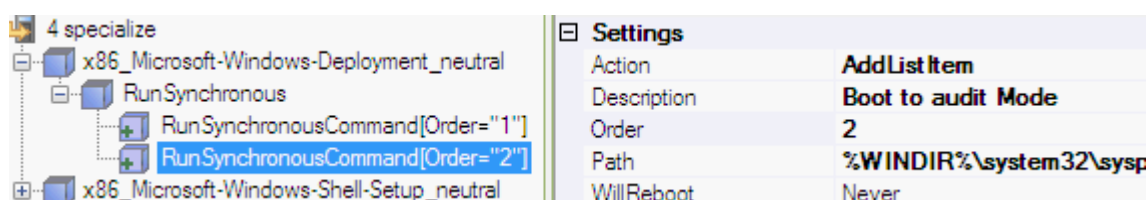
specialize \ Windows-Deployment_neutral \ RunSynchronousCommand 1



Settings	
Action	AddList Item
Description	
Order	1
Path	cmd /c "FOR %i IN (C D E F G H I J K L N M O P Q R S T U V W X Y Z) DO IF EXIST %i:\AppsRoot.txt SETX AppsRoot %i: -m"
WillReboot	Never

```
cmd /c "FOR %i IN (C D E F G H I J K L N M O P Q R S T U V W X Y Z) DO IF EXIST %i:\AppsRoot.txt SETX AppsRoot %i: -m"
```

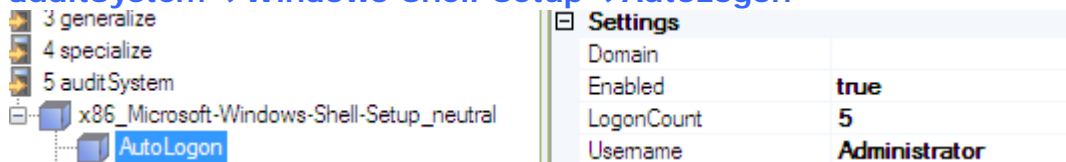
specialize \ Windows-Deployment_neutral \ RunSynchronousCommand 2



Settings	
Action	AddList Item
Description	Boot to audit Mode
Order	2
Path	%WINDIR%\system32\sysprep\sysprep.exe /quiet /audit
WillReboot	Never

```
%WINDIR%\system32\sysprep\sysprep.exe /quiet /audit
```

auditSystem \ Windows-Shell-Setup \ AutoLogon



Settings	
Domain	
Enabled	true
LogonCount	5
Username	Administrator

IMPORTANT:

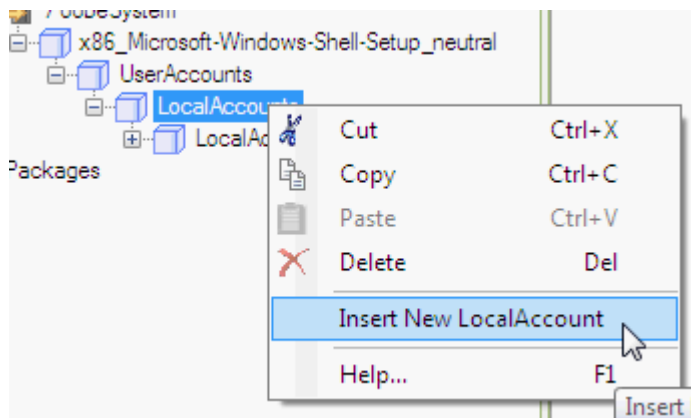
It's important to **log on** the **main administrator** account (**Administrator**) here **to avoid problems** importing registry tweaks and installing applications.

Step 3: Adding additional Settings

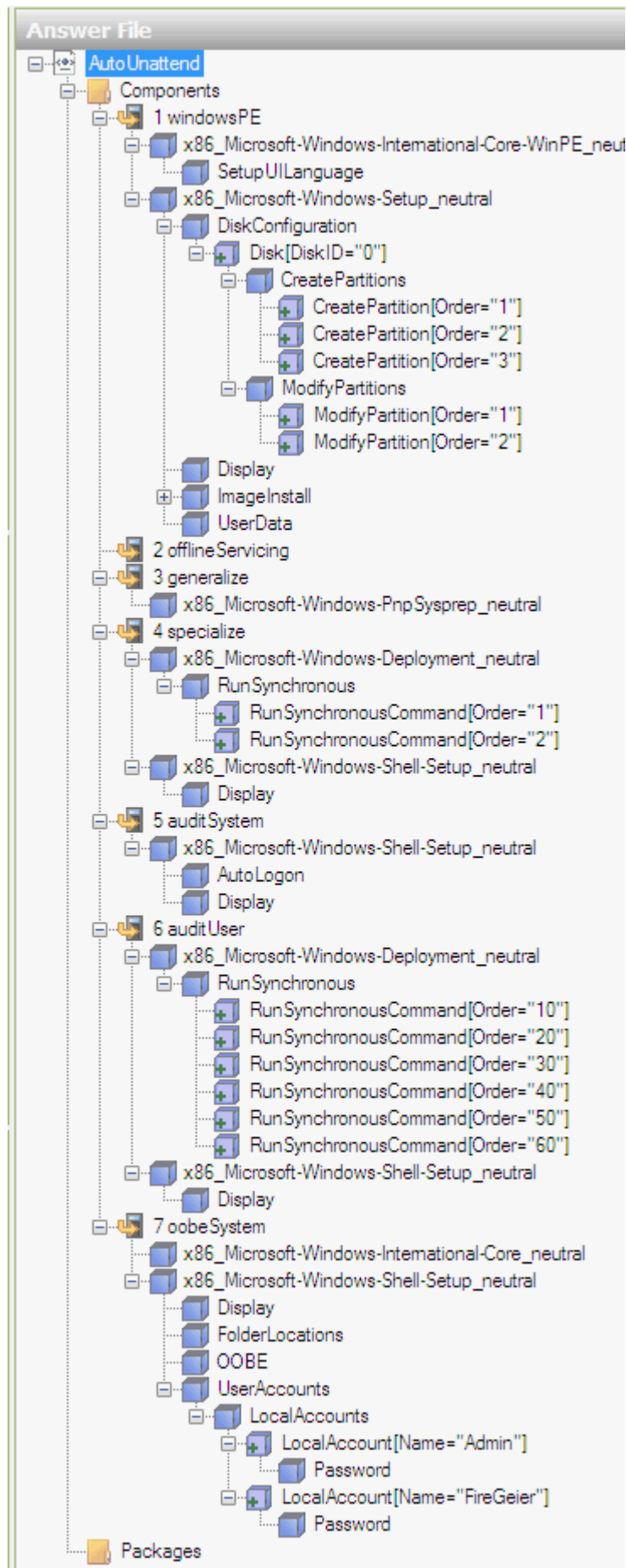
We **add** the following **components** to Autounattend.xml:

Component	Pass
Microsoft-Windows-PnPSysprep	-- Pass 3 generalize >
Microsoft-Windows-Security-Licensing-SLC-UX	-- Pass 4 specialize >
Microsoft-Windows-Shell-Setup	-- Pass 4 specialize >
Microsoft-Windows-International-Core	-- Pass 7 > oobeSystem
Microsoft-Windows-Shell-Setup\LocalAccounts\LocalAccount	-- Pass 7 oobe > System

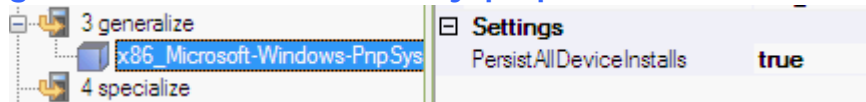
After adding component **Microsoft-Windows-Shell-Setup\LocalAccounts\LocalAccount**, we right click **LocalAccounts** in **Answer File pane** and choose **Insert New LocalAccount**:



So we get the **following structure** in Answer File pane:



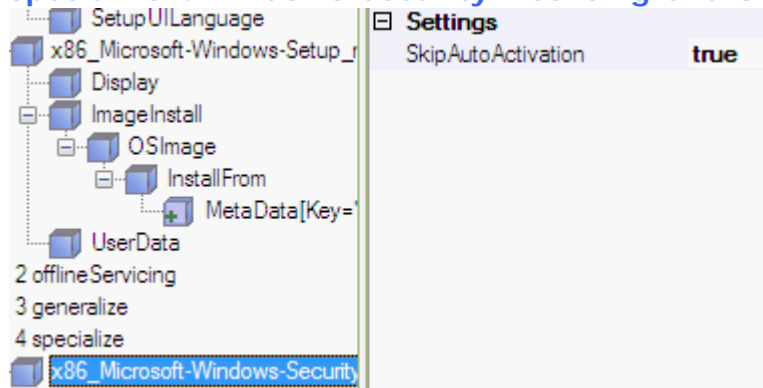
generalize \ Windows-PnP Sysprep



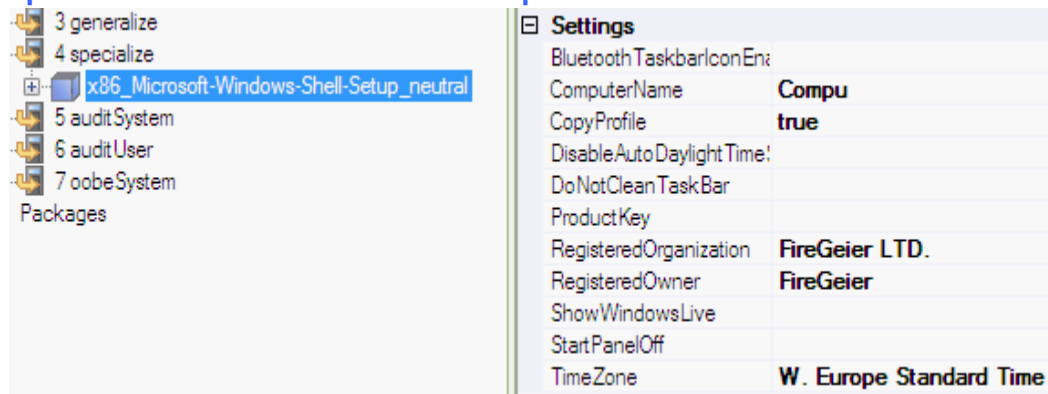
EXPLANATION:

During **sysprep generalization** all **machine specific settings** will be **removed**. So the drivers would get lost as well. **PersistAllDeviceInstalls** on **true** will **keep the drivers** so they don't need to be installed a second time.

specialize \ Windows-Security-Licensing-SLC-UX



specialize \ Windows-Shell-Setup



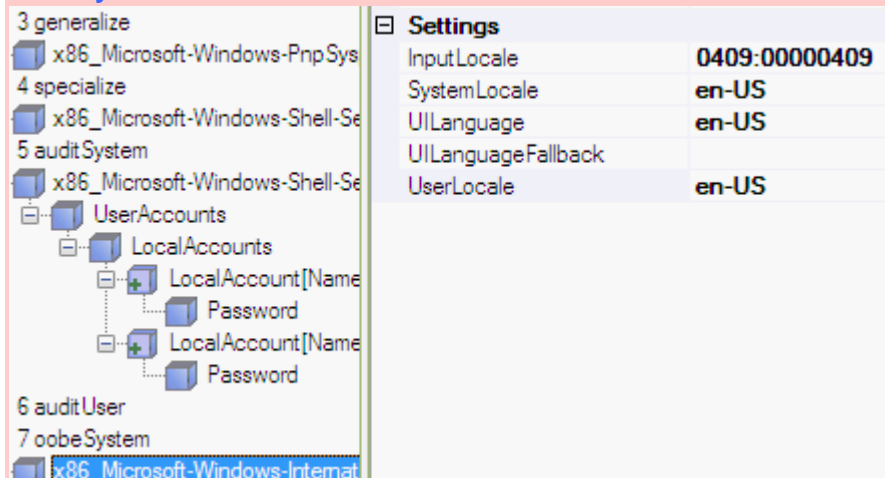
EXPLANATION:

CopyProfile set to **true** will **dispose setup** to **copy** the actual user profile to the **default user profile**.

IMPORTANT:

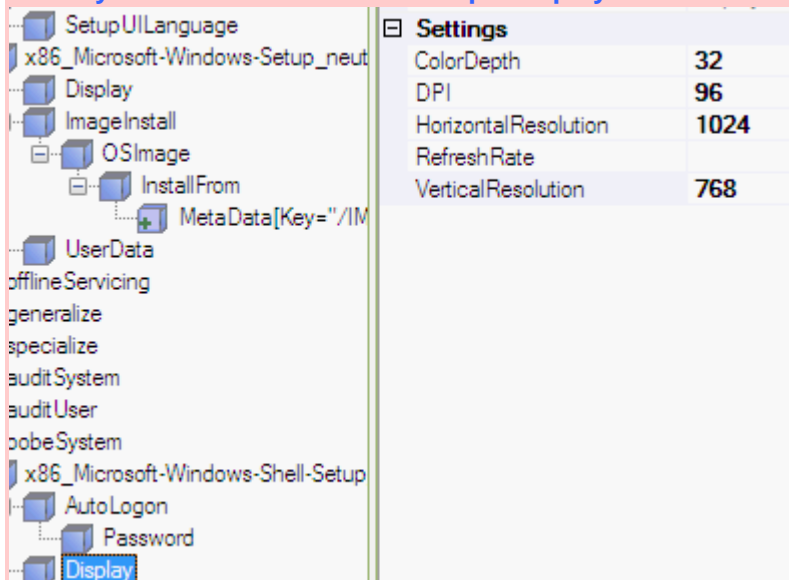
We ensure that **all settings listed in the following** are present in our Autounattend.xml. The settings are necessary to **suppress oobe** screens from popping up!

oobeSystem \ Windows-International-Core



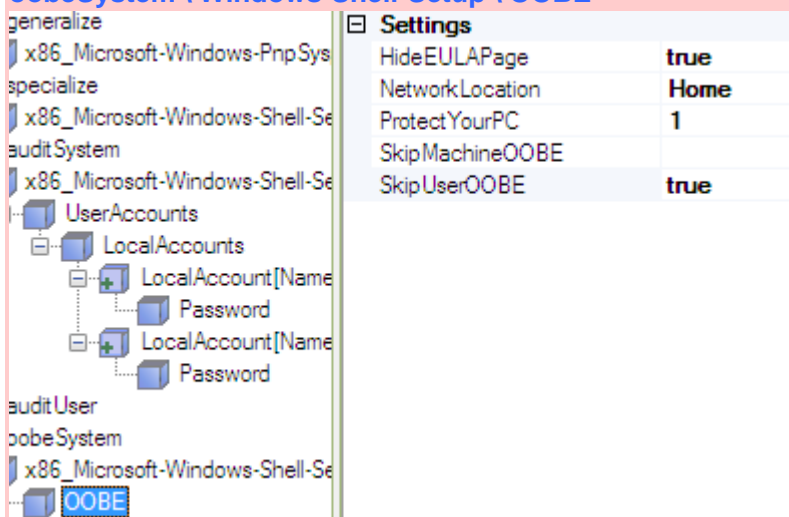
Settings	
InputLocale	0409:00000409
SystemLocale	en-US
UILanguage	en-US
UILanguageFallback	
UserLocale	en-US

oobeSystem \ Windows-Shell-Setup \ Display



Settings	
ColorDepth	32
DPI	96
HorizontalResolution	1024
RefreshRate	
VerticalResolution	768

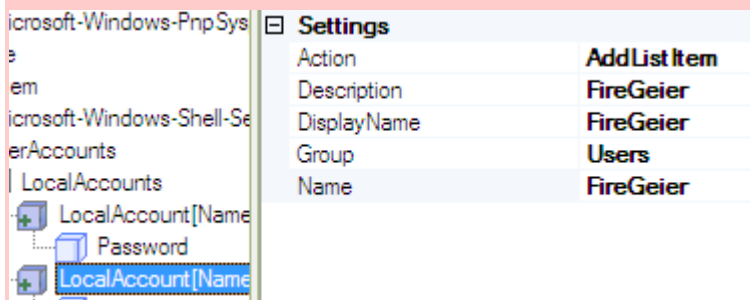
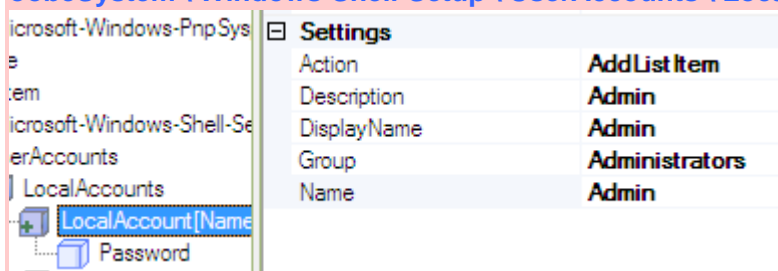
oobeSystem \ Windows-Shell-Setup \ OOBE



Settings	
HideEULAPage	true
NetworkLocation	Home
ProtectYourPC	1
SkipMachineOOBE	
SkipUserOOBE	true

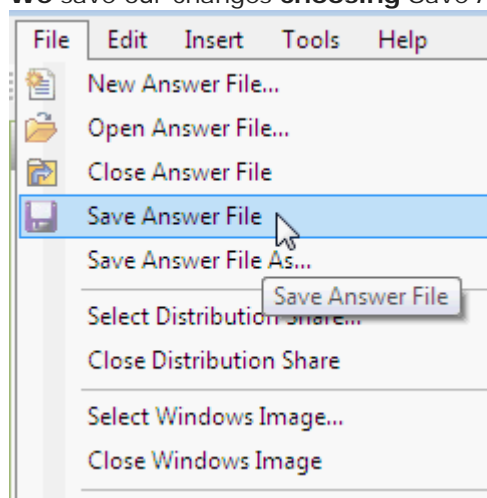
IMPORTANT:

We ensure to keep <SkipMachineOOBE> empty!

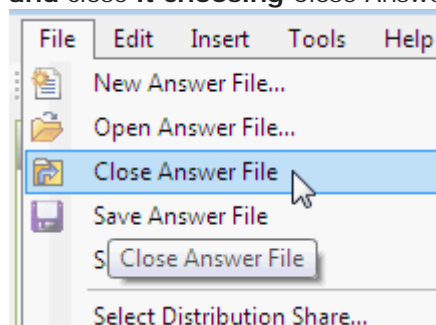
oobeSystem \ Windows-Shell-Setup \ UserAccounts \ LocalAccounts**IMPORTANT:**

We have to create **at least one Administrator account** (here Admin). The main Administrator account, we used before will be deactivated again while copying profile.

We save our changes **choosing** Save Answer File **from** File menu



and close it **choosing** Close Answer File **from** File menu:



We are finished and can now use removable media together with Vista DVD to let run setup!