# Windows

- Windows Update KB5034441 Fehler 0x80070643
- ExecutionPolicy um Skripte auszuführen
- Lange Pfad- & Dateinamen aktivieren
- Windows 11 Anmeldung umgehen
- Custom Windows11.iso
    - Einführung
    - DISM: .iso > .wim
- RDP: Kennwort ändern verhindern

# Windows Update KB5034441 Fehler 0x80070643

2024-01 Sicherheitsupdate für Windows 10 Version 22H2 für x64-basierte Systeme (KB5034441)

bei Fehlermeldung 0x80070643 muss die WinRE-Partition vergrößert werden. Microsoft ist der Fehler bekannt, sagt aber man soll es selbst beheben:

https://www.golem.de/news/windows-10-update-kb5034441-microsoft-sagt-fix-gegen-fehler-0x80070643-ab-2405-184751.html

https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/add-update-to-winre?view=windows-11#extend-the-windows-re-partition

## Zuerst ein Powershell Skript mit folgendem Inhalt (kopiert aus dem learn.microsoft Link) erstellen

zb C:\Users\Benutzer1\Documents\WindowsPowerShell\Scripts\extendWinRE.ps1

```
Param (
[Parameter(Mandatory=$false,HelpMessage="Skip confirmation")][bool]$SkipConfirmation=$false,
[Parameter(Mandatory=$true,HelpMessage="Path to backup old WinRE partition content to")][string]$BackupFolder
)
# ------------------------------------
# Helper functions
# ------------------------------------
# Log message
function LogMessage([string]$message)
{
$message = "$message"
Write-Host $message
}
```

```
# Extract numbers from string
function ExtractNumbers([string]$str)
{
    $cleanString = $str -replace "[^0-9]"
    return [long]$cleanString
}
# Display partition info using fsutil
# Return an array, the first value is total size and the second value is free space
function DisplayPartitionInfo([string[]]$partitionPath)
{
    $volume = Get-WmiObject -Class Win32_Volume | Where-Object { $partitionPath -contains
$_.DeviceID }
    LogMessage("  Partition capacity: " +  $volume.Capacity)
    LogMessage("  Partition free space: " + $volume.FreeSpace)
    return $volume.Capacity, $volume.FreeSpace
}
# Display WinRE status
function DisplayWinREStatus()
{
    # Get WinRE partition info
    $WinREInfo = Reagentc /info
    foreach ($line in $WinREInfo)
    {
        $params = $line.Split(':')
        if ($params.Count -lt 2)
        {
            continue
        }
        if (($params[1].Trim() -ieq "Enabled") -Or (($params[1].Trim() -ieq "Disabled")))
        {
            $Status = $params[1].Trim() -ieq "Enabled"
            LogMessage($line.Trim())
        }
        if ($params[1].Trim() -like "\\?\GLOBALROOT*")
        {
            $Location = $params[1].Trim()
            LogMessage($line.Trim())
        }
    }
    
```

```
    return $Status, $Location
}
# ------------------------------------
# Main execution
# ------------------------------------
# Clear the error
$Error.Clear()
# ------------------------------------
# Examining the system to collect required info
# for the execution
# Need to check WinRE status, collect OS and WinRE
# partition info
# ------------------------------------
LogMessage("Start time: $([DateTime]::Now)")
LogMessage("Examining the system...")
$NeedShrink = $true
$NeedCreateNew = $false
$NeedBackup = $false
# Get WinRE partition info
$InitialWinREStatus = DisplayWinREStatus
$WinREStatus = $InitialWinREStatus[0]
$WinRELocation = $InitialWinREStatus[1]
if (!$WinREStatus)
{
    LogMessage("Error: WinRE Disabled")
    exit 1
}
# Get System directory and ReAgent xml file
$system32Path = [System.Environment]::SystemDirectory
LogMessage("System directory: " + $system32Path)
$ReAgentXmlPath = Join-Path -Path $system32Path -ChildPath "\Recovery\ReAgent.xml"
LogMessage("ReAgent xml: " + $ReAgentXmlPath)
if (!(Test-Path  $ReAgentXmlPath))
{
    LogMessage("Error: ReAgent.xml cannot be found")
    exit 1
}
# Get OS partition
LogMessage("")
LogMessage("Collecting OS and WinRE partition info...")
```

```
$OSDrive = $system32Path.Substring(0,1)
$OSPartition = Get-Partition -DriveLetter $OSDrive
# Get WinRE partition
$WinRELocationItems = $WinRELocation.Split('\\')
foreach ($item in $WinRELocationItems)
{
    if ($item -like "harddisk*")
    {
        $OSDiskIndex = ExtractNumbers($item)
    }
    if ($item -like "partition*")
    {
        $WinREPartitionIndex = ExtractNumbers($item)
    }
}
LogMessage("OS Disk: " + $OSDiskIndex)
LogMessage("OS Partition: " + $OSPartition.PartitionNumber)
LogMessage("WinRE Partition: " + $WinREPartitionIndex)
$WinREPartition = Get-Partition -DiskNumber $OSDiskIndex -PartitionNumber $WinREPartitionIndex
$diskInfo = Get-Disk -number $OSDiskIndex
$diskType = $diskInfo.PartitionStyle
LogMessage("Disk PartitionStyle: " + $diskType)
# Display WinRE partition size info
LogMessage("WinRE partition size info")
$WinREPartitionSizeInfo = DisplayPartitionInfo($WinREPartition.AccessPaths)
LogMessage("WinRE Partition Offset: " + $WinREPartition.Offset)
LogMessage("WinRE Partition Type: " + $WinREPartition.Type)
LogMessage("OS partition size: " + $OSPartition.Size)
LogMessage("OS partition Offset: " + $OSPartition.Offset)
$OSPartitionEnds = $OSPartition.Offset + $OSPartition.Size
LogMessage("OS partition ends at: " + $OSPartitionEnds)
LogMessage("WinRE partition starts at: " + $WinREPartition.Offset)
$WinREIsOnSystemPartition = $false
if ($diskType -ieq "MBR")
{
    if ($WinREPartition.IsActive)
    {
        LogMessage("WinRE is on System partition")
        $WinREIsOnSystemPartition = $true
    }
```

```
}
if ($diskType -ieq "GPT")
{
    if ($WinREPartition.Type -ieq "System")
    {
        LogMessage("WinRE is on System partition")
        $WinREIsOnSystemPartition = $true
    }
}
# Checking the BackupFolder parameter
if ($PSBoundParameters.ContainsKey('BackupFolder'))
{
    LogMessage("")
    LogMessage("Backup Directory: [" + $BackupFolder + "]")

    $Needbackup = $true

    if ($WinREIsOnSystemPartition)
    {
        $Needbackup = $false
        LogMessage("WinRE is on System partition which will be preserved. No need to backup content")
    }
    else
    {
        if (Test-path $BackupFolder)
        {
            $items = Get-ChildItem -Path $BackupFolder
            if ($items)
            {
                LogMessage("Error: Existing backup directory is not empty")
                exit 1
            }
        }
        else
        {
            LogMessage("Creating backup directory...")
            try
            {
                $item = New-Item -Path $BackupFolder -ItemType Directory -ErrorAction Stop
                if ($item)
```

```
        {
            LogMessage("Backup directory created")
        }
        else
        {
            LogMessage("Error: Failed to create backup directory [" + $BackupFolder + "]")
            exit 1
        }
    } catch
    {
        LogMessage("Error: An error occurred: $_")
        exit 1
    }
  }
}
# ------------------------------------
# Verify whether we meet requirements of execution
# - WinRE cannot be on OS partition for the extension
# - WinRE partition must be the next partition after OS partition
# - If WinRE partition already have >=250MB free space, no need to do repartition
# - If there is enough unallocated space to grow the WinRE partition size, skip shrinking OS
#
# However, if the WinRE partition is before the OS partition, there is no chance to extend it
# As a result, it's better to create a new WinRE partition after the OS partition
# ------------------------------------
# Perform a few checks
LogMessage("")
LogMessage("Verifying if the WinRE partition needs to be extended or not...")
if (!(($diskType -ieq "MBR") -Or ($diskType -ieq "GPT")))
{
  LogMessage("Error: Got an unexpected disk partition style: " +$diskType)
  exit 1
}
# WinRE partition must be after OS partition for the repartition
if ($WinREPartitionIndex -eq $OSPartition.PartitionNumber)
{
  LogMessage("WinRE and OS are on the same partition, should not perform extension")
  exit 0
}
```

```
$supportedSize = Get-PartitionSupportedSize -DriveLetter $OSDrive
# if there is enough free space, skip extension
if ($WinREPartitionSizeInfo[1] -ge 250MB)
{
    LogMessage("More than 250 MB of free space was detected in the WinRE partition, there is no need to extend the partition")
    exit 0
}
if ($WinREPartition.Offset -lt $OSPartitionEnds)
{
    LogMessage("WinRE partition is not after OS partition, cannot perform extension")
    LogMessage("Need to create a new WinRE partition after OS partition")
    $NeedCreateNew = $true
    $NeedShrink = $true

    # Calculate the size of repartition
    # Will create a new WinRE partition with current WinRE partition size + 250 MB
    # The OS partition size will be shrunk by the new WinRE partition size
    $targetWinREPartitionSize = $WinREPartitionSizeInfo[0] + 250MB
    $shrinkSize = [Math]::Ceiling($targetWinREPartitionSize / 1MB) * 1MB
    $targetOSPartitionSize = $OSPartition.Size - $shrinkSize
    if ($targetOSPartitionSize -lt $supportedSize.SizeMin)
    {
        LogMessage("Error: The target OS partition size after shrinking is smaller than the supported minimum size, cannot perform the repartition")
        exit 1
    }
}
else
{
    if ($WinREIsOnSystemPartition)
    {
        LogMessage("WinRE parititon is after the OS partition and it's also System partition")
        LogMessage("Error: Got unexpected disk layout, cannot proceed")
        exit 1
    }
    if (!($WinREPartitionIndex -eq ($OSPartition.PartitionNumber + 1)))
    {
        LogMessage("Error: WinRE partition is not right after the OS partition, cannot extend WinRE partition")
```

```
    exit 1
  }
  # Calculate the size of repartition
  # Will shrink OS partitition by 250 MB
  $shrinkSize = 250MB
  $targetOSPartitionSize = $OSPartition.Size - $shrinkSize
  $targetWinREPartitionSize = $WinREPartitionSizeInfo[0] + 250MB
  $UnallocatedSpace = $WinREPartition.Offset - $OSPartitionEnds;
  # If there is unallocated space, consider using it
  if ($UnallocatedSpace -ge 250MB)
  {
    $UnallocatedSpace = $WinREPartition.Offset - $OSPartitionEnds;
    LogMessage("Found unallocated space between OS and WinRE partition: " + $UnallocatedSpace)
    LogMessage("There is already enough space to extend WinRE partition without shrinking the OS
partition")
    $NeedShrink = $false
    $targetOSPartitionSize = 0
  }
  else
  {
    $shrinkSize = [Math]::Ceiling((250MB - $UnallocatedSpace)/ 1MB) * 1MB
    if ($shrinkSize > 250MB)
    {
      $shrinkSize = 250MB
    }
    $targetOSPartitionSize = $OSPartition.Size - $shrinkSize
    if ($targetOSPartitionSize -lt $supportedSize.SizeMin)
    {
      LogMessage("Error: The target OS partition size after shrinking is smaller than the supporte
minimum size, cannot perform the repartition")
      exit 1
    }
  }
}
# ----------------------------------
# Report execution plan and ask for user confirmation to continue
# ----------------------------------
# Report the changes planned to be executed, waiting for user confirmation
LogMessage("")
LogMessage("Summary of proposed changes")
```

```
if ($NeedCreateNew)
{
    LogMessage("Note: WinRE partition is before OS partition, need to create a new WinRE partition
after OS partition")
    LogMessage("Will shrink OS partition by " + $shrinkSize)
    LogMessage("  Current OS partition size: " + $OSPartition.Size)
    LogMessage("  Target OS partition size after shrinking: " + $targetOSPartitionSize)
    LogMessage("New WinRE partition will be created with size: ", $targetWinREPartitionSize)
    if ($WinREIsOnSystemPartition)
    {
        LogMessage("Existing WinRE partition is also system partition, it will be preserved")
    }
    else
    {
        LogMessage("Existing WinRE partition will be deleted")
        LogMessage("  WinRE partition: Disk [" + $OSDiskIndex + "] Partition [" + $WinREPartitionInde
+ "]")
        LogMessage("  Current WinRE partition size: " + $WinREPartitionSizeInfo[0])
    }
}
else
{
    if ($NeedShrink)
    {
        LogMessage("Will shrink OS partition by " + $shrinkSize)
        LogMessage("  Current OS partition size: " + $OSPartition.Size)
        LogMessage("  Target OS partition size after shrinking: " + $targetOSPartitionSize)
        if ($UnallocatedSpace -ge 0)
        {
            LogMessage("Unallocated space between OS and WinRE partition that will be used towards the n
WinRE partition: " + $UnallocatedSpace)
        }
    }
    else
    {
        LogMessage("Will use 250MB from unallocated space between OS and WinRE partition")
    }
    LogMessage("Will extend WinRE partition size by 250MB")
    LogMessage("  WinRE partition: Disk [" + $OSDiskIndex + "] Partition [" + $WinREPartitionIndex
+ "]")
```

```
    LogMessage("  Current WinRE partition size: " + $WinREPartitionSizeInfo[0])
    LogMessage("  New WinRE partition size:     " + $targetWinREPartitionSize)
    LogMessage("WinRE will be temporarily disabled before extending the WinRE partition and
enabled automatically in the end")
    if ($UnallocatedSpace -ge 100MB)
    {
        LogMessage("Warning: More than 100MB of unallocated space was detected between the OS and
WinRE partitions")
        LogMessage("Would you like to proceed by using the unallocated space between the OS and the
WinRE partitions?")
    }
}
if ($Needbackup)
{
    LogMessage("")
    LogMessage("The contents of the old WinRE partition will be backed up to [" + $BackupFolder +
"]")
}
LogMessage("")
LogMessage("Please reboot the device before running this script to ensure any pending
partition actions are finalized")
LogMessage("")
if ($SkipConfirmation)
{
    LogMessage("User chose to skip confirmation")
    LogMessage("Proceeding with changes...")
}
else
{
    $userInput = Read-Host -Prompt "Would you like to proceed? Y for Yes and N for No"

    if ($userInput -ieq "Y")
    {
        LogMessage("Proceeding with changes...")
    }
    elseif ($userInput -ieq "N")
    {
        LogMessage("Canceling based on user request, no changes were made to the system")
        exit 0
    }
```

```
else
{
    LogMessage("Error: Unexpected user input: [" + $userInput + "]")
    exit 0
}
}
LogMessage("")
LogMessage("Note: To prevent unexpected results, please do not interrupt the execution or restart your system")
# ----------------------------------
# Do the actual execution
# The main flow is:
# 1. Check whether ReAgent.xml has stage location and clear it for repartiion
# 2. Disable WinRE as WinRE partition will be deleted
# 3. Perform the repartition to create a larger WinRE partition
# 4. Re-enable WinRE
# ----------------------------------
LogMessage("")
# Load ReAgent.xml to clear Stage location
LogMessage("Loading [" + $ReAgentXmlPath + "] ...")
$xml = [xml](Get-Content -Path $ReAgentXmlPath)
$node = $xml.WindowsRE.ImageLocation
if (($node.path -eq "") -And ($node.guid -eq "{00000000-0000-0000-0000-000000000000}") -And ($node.offset -eq "0") -And ($node.id -eq "0"))
{
    LogMessage("Stage location info is empty")
}
else
{
    LogMessage("Clearing stage location info...")
    $node.path = ""
    $node.offset = "0"
    $node.guid= "{00000000-0000-0000-0000-000000000000}"
    $node.id="0"
    # Save the change
    LogMessage("Saving changes to [" + $ReAgentXmlPath + "]...")
    $xml.Save($ReAgentXmlPath)
}
# Disable WinRE
LogMessage("Disabling WinRE...")
```

```
reagentc /disable
if (!($LASTEXITCODE -eq 0))
{
    LogMessage("Warning: encountered an error when disabling WinRE: " + $LASTEXITCODE)
    exit $LASTEXITCODE
}
# Verify WinRE is under C:\Windows\System32\Recovery\WinRE.wim
$disableWinREPath = Join-Path -Path $system32Path -ChildPath "\Recovery\WinRE.wim"
LogMessage("Verifying that WinRE wim exists in downlevel at default location")
if (!(Test-Path $disableWinREPath))
{
    LogMessage("Error: Cannot find " + $disableWinREPath)
    
    # Re-enable WinRE
    LogMessage("Re-enabling WinRE on error...")
    reagentc /enable
    if (!($LASTEXITCODE -eq 0))
    {
        LogMessage("Warning: encountered an error when enabling WinRE: " + $LASTEXITCODE)
    }
    exit 1
}
# -----------------------------------
# Perform the repartition
# 1. Resize the OS partition
# 2. Delete the WinRE partition
# 3. Create a new WinRE partition
# -----------------------------------
LogMessage("Performing repartition to extend the WinRE partition ...")
# 1. Resize the OS partition
if ($NeedShrink)
{
    LogMessage("Shrinking the OS partition to create a larger WinRE partition")
    LogMessage("Resizing the OS partition to: [" + $targetOSPartitionSize + "]...")
    Resize-Partition -DriveLetter $OSDrive -Size $targetOSPartitionSize
    if ($Error.Count -gt 0) {
        LogMessage("Error: Resize-Partition encountered errors: " + $Error[0].Exception.Message)
        
        # Re-enable WinRE
        LogMessage("Re-enabling WinRE on error...")
```

```
    reagentc /enable
    if (!($LASTEXITCODE -eq 0))
    {
        LogMessage("Warning: encountered an error when enabling WinRE: " + $LASTEXITCODE)
    }
    exit 1
}
$OSPartitionAfterShrink = Get-Partition -DriveLetter $OSDrive
LogMessage("Target partition size: " + $targetOSPartitionSize)
LogMessage("Size of OS partition after shrinking: " + $OSPartitionAfterShrink.Size)
}
# 2. Delete the WinRE partition
LogMessage("")
if ($WinREIsOnSystemPartition)
{
LogMessage("Existing WinRE partition is System partition, skipping deletion")
}
else
{
# If requested by user, backup rest of the content on WinRE partition to backup directory
if ($Needbackup)
{
    $sourcePath = $WinREPartition.AccessPaths[0]
    LogMessage("Copying content on WinRE partition from [" + $sourcePath + "] to [" +
$BackupFolder + "]...")

    # Copy-Item may have access issue with certain system folders, enumerate the children items
and exlcude them
    $items = Get-ChildItem -LiteralPath $sourcePath -Force
    foreach ($item in $items)
    {
        if ($item.Name -ieq "System Volume Information")
        {
            continue
        }
        $sourceItemPath = Join-Path -Path $sourcePath -ChildPath $item.Name
        $destItemPath = Join-Path -Path $BackupFolder -ChildPath $item.Name
        try
        {
            LogMessage("Copying [" + $sourceItemPath + "] to [" + $destItemPath + "]...")
```

```
        Copy-Item -LiteralPath $sourceItemPath -Destination $destItemPath -Recurse -Force
    } catch
    {
        LogMessage("Error: An error occurred during copy: $_")
        exit 1
    }
  }

  LogMessage("Backup completed")
  LogMessage("")
 }
 LogMessage("Deleting WinRE partition: Disk [" + $OSDiskIndex + "] Partition [" +
$WinREPartitionIndex + "]...")
 Remove-Partition -DiskNumber $OSDiskIndex -PartitionNumber $WinREPartitionIndex -
Confirm:$false
 if ($Error.Count -gt 0) {
  LogMessage("Error: Remove-Partition encountered errors: " + $Error[0].Exception.Message)
  exit 1
 }
}
# A short sleep for the partition change
Sleep 5
# 3. Create a new WinRE partition
LogMessage("")
LogMessage("Creating new WinRE partition...")
LogMessage("Target size: " + $targetWinREPartitionSize)
if ($diskType -ieq "GPT")
{
 $partition = New-Partition -DiskNumber $OSDiskIndex -Size $targetWinREPartitionSize -GptType
"{de94bba4-06d1-4d40-a16a-bfd50179d6ac}"

 $newPartitionIndex = $partition.PartitionNumber
 # A short sleep to make sure the partition is ready for formatting
 Sleep 2
 LogMessage("Formating the partition...")
 $result = Format-Volume -Partition $partition -FileSystem NTFS -Confirm:$false
 if ($Error.Count -gt 0) {
  LogMessage("Error: Format-Volume encountered errors: " + $Error[0].Exception.Message)
  exit 1
 }
```

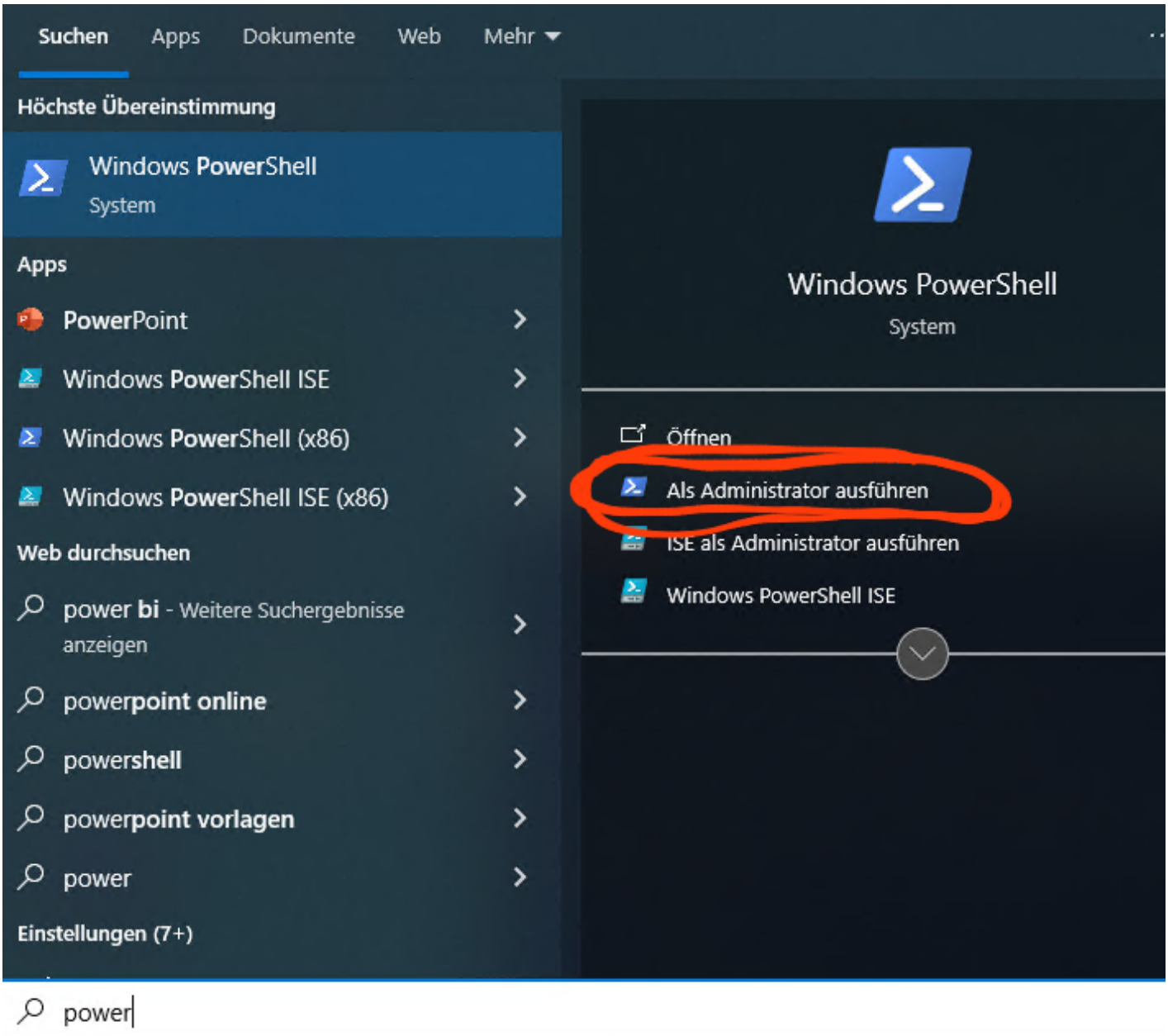
```
}
else
{
□#$partition = New-Partition -DiskNumber $OSDiskIndex -Size $targetWinREPartitionSize -MbrType
0x27
□$targetWinREPartitionSizeInMb = [int]($targetWinREPartitionSize/1MB)
□$diskpartScript =
@"
select disk $OSDiskIndex
create partition primary size=$targetWinREPartitionSizeInMb id=27
format quick fs=ntfs label="Recovery"
set id=27
"@
□$TempPath = $env:Temp
□$diskpartSciptFile = Join-Path -Path $TempPath -ChildPath
"\ExtendWinRE_MBR_PowershellScript.txt"
□
□LogMessage("Creating temporary diskpart script to create Recovery partition on MBR disk...")
□LogMessage("Temporary diskpart script file: " + $diskpartSciptFile)
□$diskpartScript | Out-File -FilePath $diskpartSciptFile -Encoding ascii
□
□LogMessage("Executing diskpart script...")
□try
□{
□□$diskpartOutput = diskpart /s $diskpartSciptFile
□□
□□if ($diskpartOutput -match "DiskPart successfully")
□□{
□□□LogMessage("Diskpart script executed successfully")
□□}
□□else
□□{
□□□LogMessage("Error executing diskpart script:" + $diskpartOutput)
□□□exit 1
□□}
□□LogMessage("Deleting temporary diskpart script file...")
□□Remove-Item $diskpartSciptFile
□}
□catch
□{
```

```
    LogMessage("Error executing diskpart script: $_")
    exit 1
}

$vol = Get-Volume -FileSystemLabel "Recovery"
$newPartitionIndex = (Get-Partition | Where-Object { $_.AccessPaths -contains $vol.Path }
).PartitionNumber
}
if ($Error.Count -gt 0)
{
LogMessage("Error: New-Partition encountered errors: " + $Error[0].Exception.Message)
exit 1
}
LogMessage("New Partition index: " + $newPartitionIndex)
# Re-enable WinRE
LogMessage("Re-enabling WinRE...")
reagentc /enable
if (!($LASTEXITCODE -eq 0))
{
LogMessage("Warning: encountered an error when enabling WinRE: " + $LASTEXITCODE)
exit $LASTEXITCODE
}
# In the end, Display WinRE status to verify WinRE is enabled correctly
LogMessage("")
LogMessage("WinRE Information:")
$FinalWinREStatus = DisplayWinREStatus
$WinREStatus = $FinalWinREStatus[0]
$WinRELocation = $FinalWinREStatus[1]
if (!$WinREStatus)
{
LogMessage("Warning: WinRE Disabled")
}
$WinRELocationItems = $WinRELocation.Split('\\')
foreach ($item in $WinRELocationItems)
{
if ($item -like "partition*")
{
    $WinREPartitionIndex = ExtractNumbers($item)
}
}
```

```
LogMessage("WinRE Partition Index: " + $WinREPartitionIndex)
$WinREPartition = Get-Partition -DiskNumber $OSDiskIndex -PartitionNumber $WinREPartitionIndex
$WinREPartitionSizeInfoAfter = DisplayPartitionInfo($WinREPartition.AccessPaths)
LogMessage("")
LogMessage("OS Information:")
$OSPartition = Get-Partition -DriveLetter $OSDrive
LogMessage("OS partition size: " + $OSPartition.Size)
LogMessage("OS partition Offset: " + $OSPartition.Offset)
if (!($WinREPartitionIndex -eq $newPartitionIndex))
{
LogMessage("Warning: WinRE is installed to partition [" + $WinREPartitionIndex +"], but the
newly created Recovery partition is [" + $newPartitionIndex + "]")
}
LogMessage("End time: $([DateTime]::Now)")
if ($NeedBackup)
{
LogMessage("")
LogMessage("The contents of the old WinRE partition has been backed up to [" + $BackupFolder +
"]")
}
LogMessage("")
LogMessage("Successfully completed the operation")
```

## dann die Powershell als Admin starten:




## dann die ExecutionPolicy ändern

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy remotesigned
```

## Skript ausführen

```
C:\Users\Benutzer1\Documents\WindowsPowerShell\Scripts\extendWinRE.ps1 -SkipConfirmation $true
-BackupFolder c:\winre_backup
```

# ExecutionPolicy um Skripte auszuführen

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy remotesigned
```

oder

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

# Lange Pfad- & Dateinamen aktivieren

## Variante 1

In PowerShell mit Adminrechten:

```
Set-ItemProperty 'HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem' -Name 'LongPathsEnabled'
-Value 1
```

Quelle:

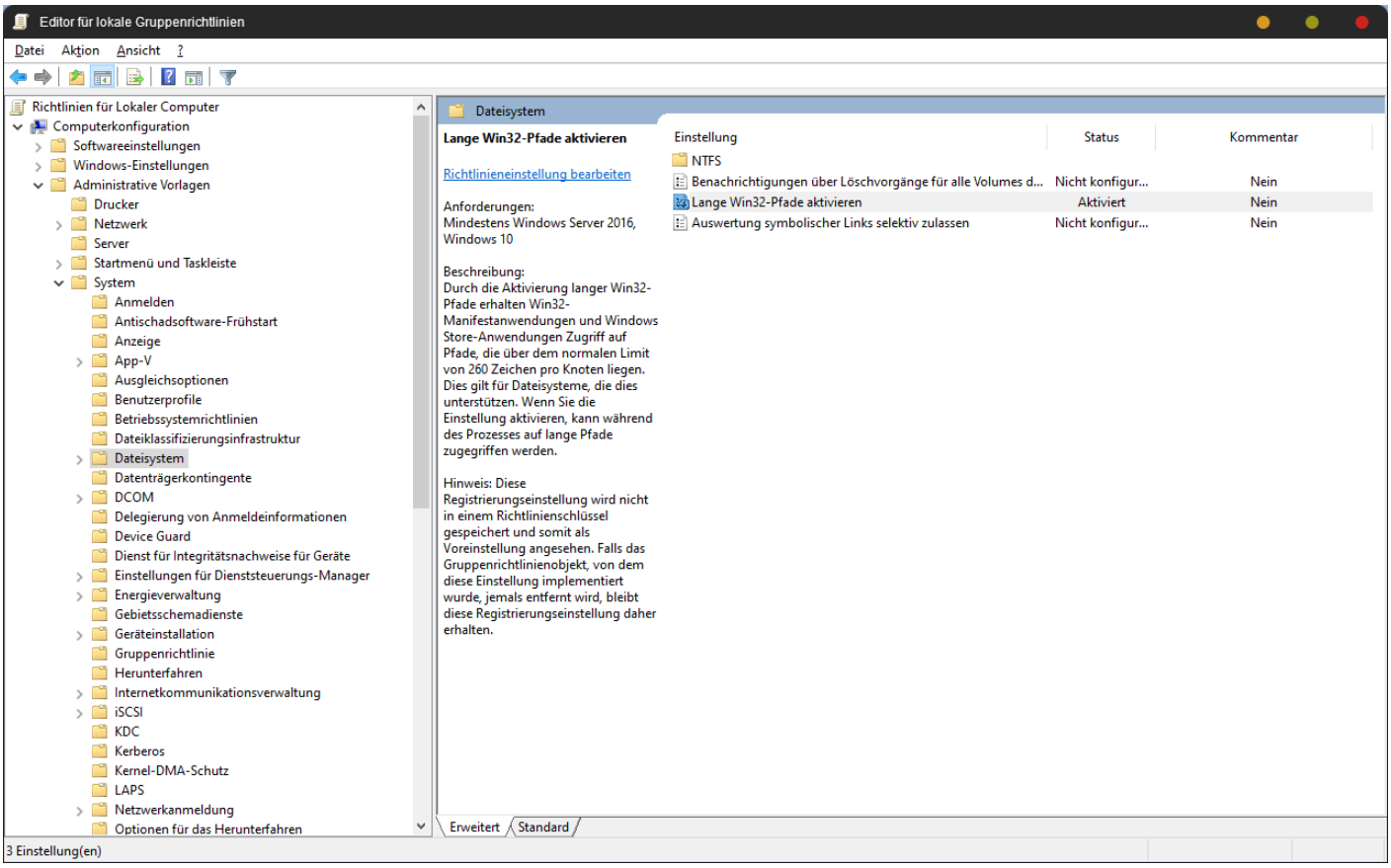
https://lgug2z.github.io/komorebi/installation.html

Stichwörter:

Lange Dateinamen, Lange Pfade

## Varienate 2

WIN+R > gpedit.msc > Computerkonfig > Administrative Vorlagen > System > Dateisystem > Lange Win32-Pfade aktivieren > Aktivieren > PC neustarten




> lange Pfade sind immer hinter einem \\?\C:\Windows\... versteckt

Alternativ die Datei in Powershell umbenennen

```
Rename-Item -LiteralPath "\\?\D:\Entwicklung\Superlangerpdfname.pdf" -NewName "test.pdf
```

oder löschen

```
Remove-Item -LiteralPath "\\?\D:\Entwicklung\Superlangerpdfname.pdf"
```

# Windows 11 Anmeldung umgehen

https://www.reddit.com/r/Windows11/comments/1c2dcoq/bypass_windows_11_microsoft_account_requirement/

Shift+F10:
`oobe\BypassNRO` eingeben
Neustart

Beim Anmelden Screen nochmal Shift+F10:
ipconfig /release
cmd schließen
Pfeil zurück
Pfeil wieder vor (weiter)

fertig

# Custom Windows11.iso

# Einführung

Dieses Kapitel beschreibt, wie man mit einer Hand voll Tools ein benutzerdefiniertes Windows 11 erstellen kann. Das Image soll folgende Features besitzen:

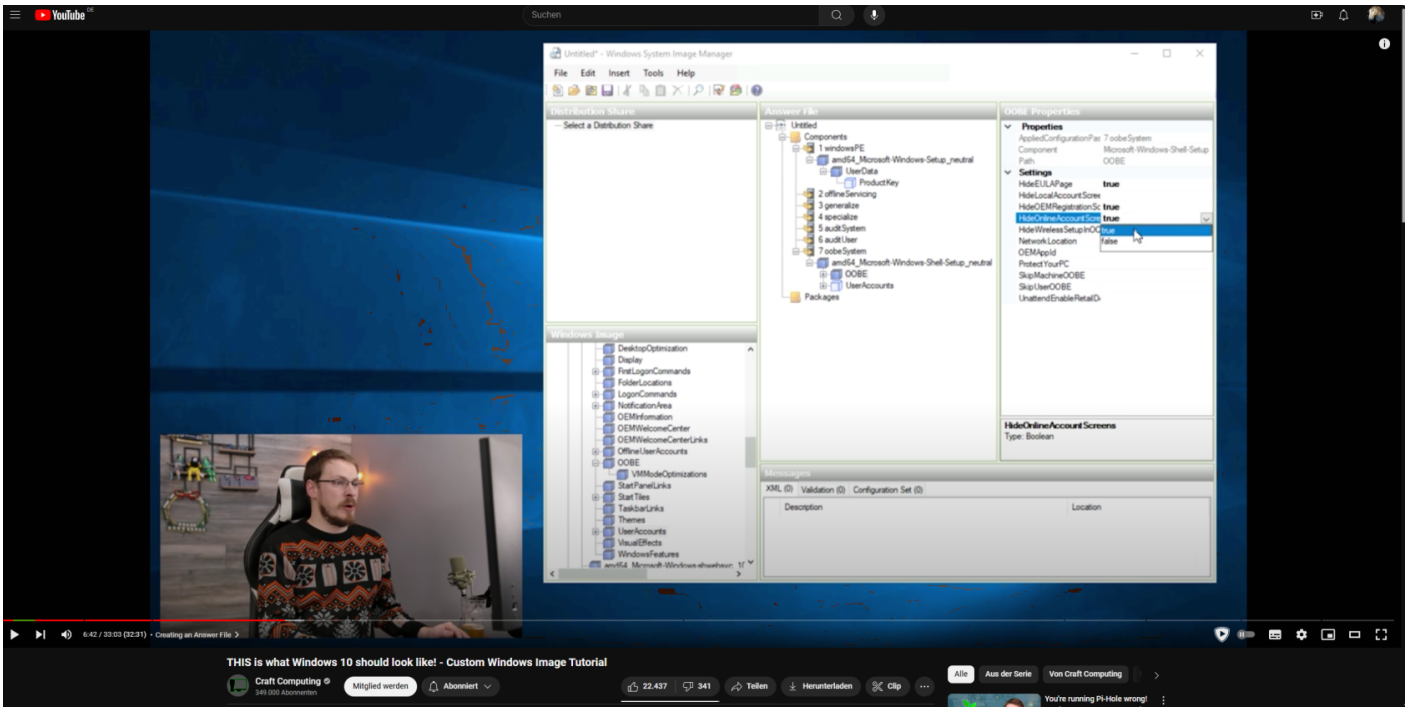
- Bootbar vom USB-Stick Ventoy
- Fast vollständig unbeaufsichtigte WIndows Installation
- Einzig die Eingabe von Benutzername, Passwort und Hostname während der Installation
- Vorinstallierte Ordner mit Dateien zb eine Installationsdatei für bestimmte Software
- Automatisches Ausführen eines Powershell Skripts nach dem ersten Boot, das Software von einem Paketmanager (Winget oder Chocolatey) installiert
- Automatisches Ausführen eines Python Skripts nach dem ersten Boot, zb um einen E-Mail Account über einen Makrobot bei Strato anzulegen, Slack anzumelden, Office zu aktivieren, Windows zu aktivieren etc.

# Tools

- Ventoy (USB-Stick mount mit einfachem ISO-Kopieren statt mit Rufus flashen) oder iVentoy für PXE netboot
- Windows ADK
  - Windows System Image Manager für Antwortdatei unattend.xml
  - DISM für .iso >.wim
- Windows PE
- Winutil
- Sysprep

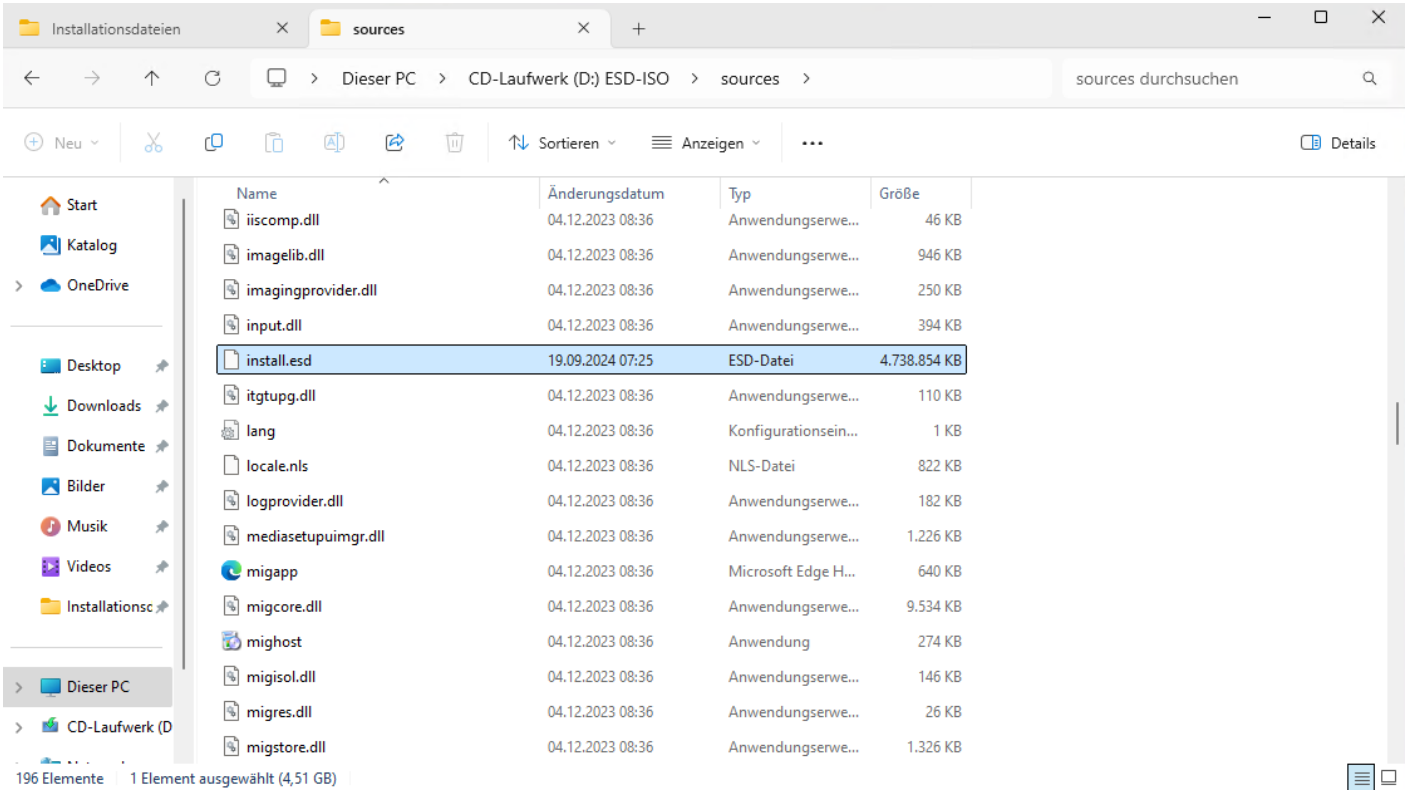
# Tutorial

https://www.youtube.com/watch?v=PdKMiFKGQuc


YouTube
Suchen
Untitled* - Windows System Image Manager
File Edit Insert Tools Help
Distribution Share
Select a Distribution Share
Answer File
Untitled
Components
1 windowsPE
amd64_Microsoft-Windows-Setup_neutral
UserData
ProductKey
2 offlineServicing
3 generalize
4 specialize
5 auditSystem
6 auditUser
7 oobeSystem
amd64_Microsoft-Windows-Shell-Setup_neutral
OOBE
UserAccounts
Packages
OOBE Properties
Properties
Settings
HideEULAPage true
HideOnlineAccountScreens true
HideWirelessSetupInOOBE true
false
HideOnlineAccountScreens
Type: Boolean
Windows Image
DesktopOptimization
Display
FirstLogonCommands
FolderLocations
LogonCommands
NotificationArea
OEMInformation
OEMWelcomeCenter
OEMWelcomeCenterLinks
OfflineUserAccounts
OOBE
VMModeOptimizations
StartPanelLinks
StartTiles
TaskbarLinks
Themes
UserAccounts
VisualEffects
WindowsFeatures
Messages
XML (0) Validation (0) Configuration Set (0)
Description
Location
6:42 / 33:03 (32:31) • Creating an Answer File
THIS is what Windows 10 should look like! - Custom Windows Image Tutorial
Craft Computing
349.000 Abonnenten
Mitglied werden
Abonniert
22.437
341
Teilen
Herunterladen
Clip
Alle
Aus der Serie
Von Craft Computing
You're running Pi-Hole wrong!

# DISM: .iso > .wim

Um an die .wim Datei zu kommen, muss man die windows11.iso Datei mounten. Dort die .esd kopieren, bei neueren Windowsversionen meistens `CD-Laufwerk:\sources\install.esd` oder dort ist direkt die .wim zu finden




## Index von Pro oder Home Variante in .esd finden

```
dism /get-wiminfo /wimfile:"C:\Pfad\zur\install.esd"
```

## .wim aus .esd extrahieren

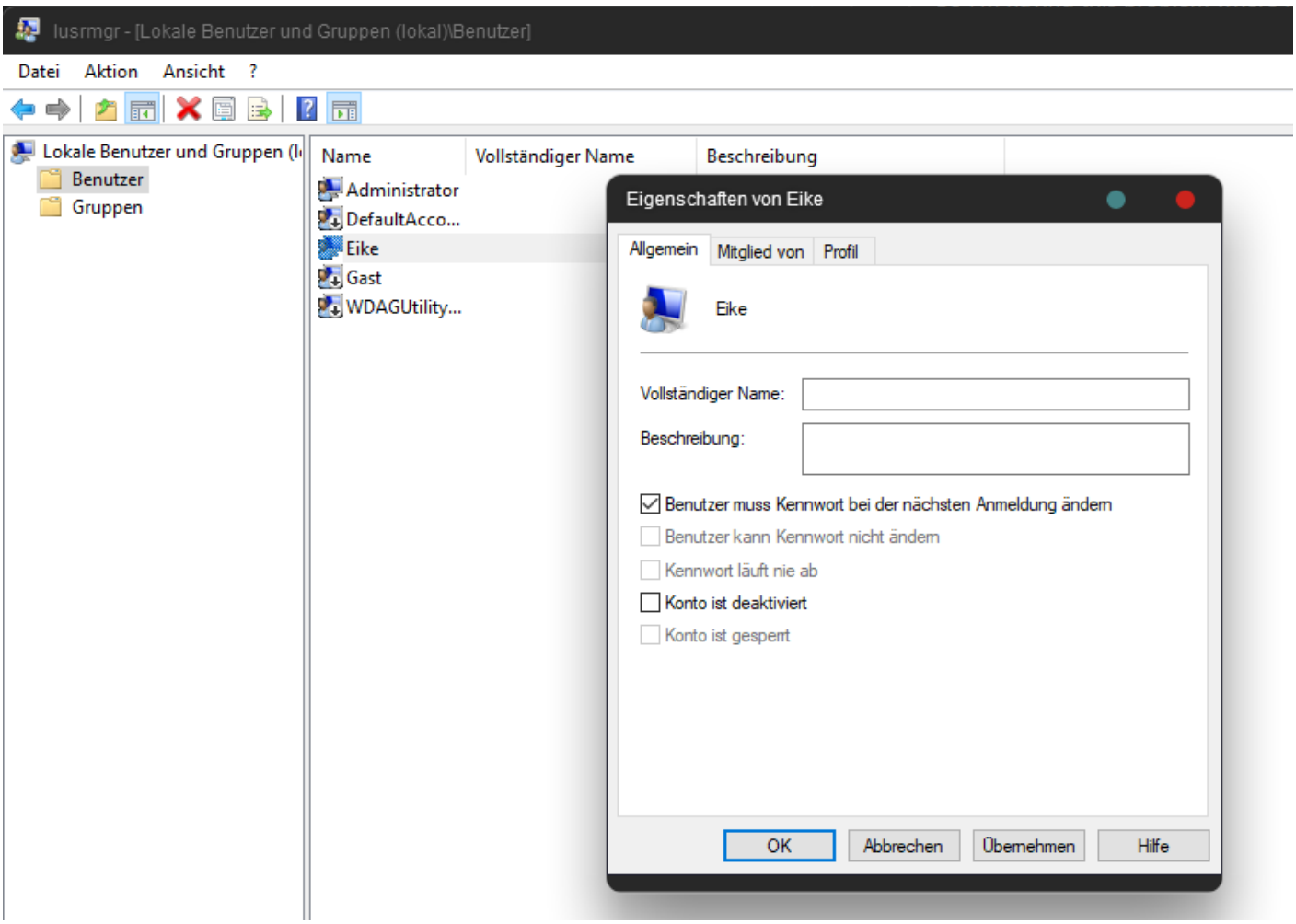
```
dism /export-image /sourceimagefile:"install.esd" /sourceindex:5
/destinationimagefile:"w11_23H2_pro.wim" /compress:max /checkintegrity
```

# RDP: Kennwort ändern verhindern

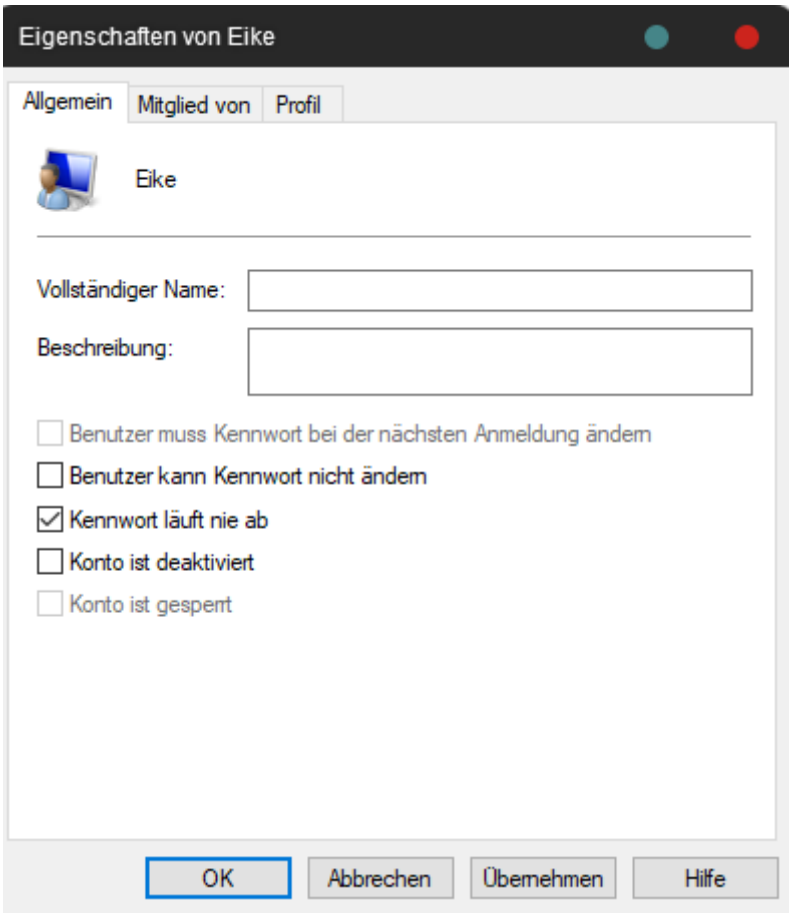
win+r (Ausführen)

```
lusrmgr.msc
```

starten. Benutzer > User auswählen:




ändern zu


Eigenschaften von Eike
Allgemein
Mitglied von
Profil
Eike
Vollständiger Name:
Beschreibung:
Benutzer muss Kennwort bei der nächsten Anmeldung ändern
Benutzer kann Kennwort nicht ändern
Kennwort läuft nie ab
Konto ist deaktiviert
Konto ist gesperrt
OK
Abbrechen
Übernehmen
Hilfe