

Windows

Yeah, Windoze is here too

- [Active Directory](#)
- [Windows Management Consoles](#)
- [Windows Install Media](#)

Active Directory

To get a list of the FSMO Role holders for a Single Domain.

```
Get-ADDomain | Select-Object DistinguishedName, SchemaMaster, DomainNamingMaster, InfrastructureMaster, PDCEmulator, RIDMaster
```

To get a list of the FSMO Role holders in a Forest.

```
Get-ADForest | Select-Object Name,SchemaMaster, DomainNamingMaster,InfrastructureMaster, PDCEmulator, RIDMasterall
```

To get a nicely formatted list with all the Domain Controllers and who owns which particular role.

```
Get-ADDomainController -Filter * | Select-Object Name, Domain, Forest, OperationMasterRoles | Where-Object {$_.OperationMasterRoles}
```

Create a new AD computer.

```
$ServerName="newsys"
$DC="ads01.example.com"
$OU="OU=SERVERS,DC=example,DC=com"
New-ADComputer -Name $ServerName -SamAccountName $ServerName -Path $OU -Description "Apache Win Development" -Server $DC -Enabled $True
```

Unlock an account

```
get-ADUser -Identity <username> -Properties LockedOut
Unlock-ADAccount -Identity <username>
```

Change a password

```
$SecPaswd= ConvertTo-SecureString -String 'kPnguoHTUQ' -AsPlainText -Force
Set-ADAccountPassword -Reset -NewPassword $SecPaswd -Identity cescvss01
Set-ADUser -Identity <username> -ChangePasswordAtLogon $false
```

New AD group

```
$GroupName="SvrOps"
```

```
$OU="OU=GROUPS,DC=exmple,DC=com"
```

```
New-ADGroup $GroupName -Path $OU -GroupCategory Security -GroupScope Global -PassThru -Verbose
```

Windows Management Consoles

CERTMGR.MSC	Certificates snap-in
CERTSRV.MSC	Certification Services
CMD.EXE	Command Prompt
COMPMGMT.MSC	Computer Management
DCPOL.MSC	Domain Controller Security Policy
DEVMGMT.MSC	Device Manager
DFRG.MSC	Disk Defragmenter
DFSGUI.MSC	Distributed File System
DHCPMGMT.MSC	DHCP Manager
DISKMGMT.MSC	Disk Management
DNSMGMT.MSC	DNS Manager
DOMAIN.MSC	Active Directory Domains & Trust
DOMPOL.MSC	Domain Security Policy
DSA.MSC	Active Directory Users & Computers
DSA.MSC /DOMAIN=domainname	
DSA.MSC /SERVER=servername	
DSSITE.MSC	Active Directory Sites & Services
EVENTVWR.MSC	Event Viewer
FSMGMT.MSC	Shared Folders
GPEDIT.MSC	local Group Policy Editor
IAS.MSC	Internet Authentication Service
INETMGR	Internet Information Service (\Windows\system32\inetsrv)
LUSRMGR.MSC	Local Users and Groups
MMC.EXE	Microsoft Management Console
MSINFO32.EXE	Hardware and software configuration information
PERFMON.MSC	Performance Monitor
REGEDIT.EXE	Run Registry Editor
RRASMGMT.MSC	Routing and Remote Access
RSOP.MSC	Resultant Set of Policy
SECPOL.MSC	Local Security Policy
SERVICES.MSC	Service Configuration
TSCC.MSC	Terminal Services
MSTSC	Remote Desktop

Windows Install Media

```
./create_win_usb.sh ~/Downloads/Win10.iso /dev/sdc
```

```
#!/bin/bash
# -----
# Create Bootable Windows 10/11 USB in Linux (Debian/RPM-based)
# Version: 1.0
# Author: LinuxConfig.org
# Date: 23-06-2023
# License: GPL-3.0
# -----
# Input parameters: Path to the ISO file and USB block device location
ISO_PATH=$1
USB_BLOCK=$2

echo "ISO Path is: $ISO_PATH"
echo "USB block device is: $USB_BLOCK"

# Check for required commands
REQUIRED_COMMANDS=("rsync" "parted" "wipefs" "mkfs.vfat" "mkfs.ntfs" "udisksctl" "sha256sum" "mount"
"umount" "mkdir" "cp" "sync" "mktemp")

echo "Checking for required commands..."
for cmd in "${REQUIRED_COMMANDS[@]}; do
    if ! command -v $cmd >/dev/null 2>&1; then
        echo "Missing command $cmd. Please install the corresponding package and rerun this script."
        exit 1
    fi
done
echo "All required commands are available."

# 1. Checking ISO file
read -p "Do you want to calculate the ISO file checksum? This could take some time. Press 'y' to continue or any
other key to skip: " response
if [[ $response =~ ^[Yy]$ ]]
then
    echo "Calculating ISO file checksum..."
```

```
ISO_CHECKSUM=$(sha256sum $ISO_PATH)
echo "Checksum is: $ISO_CHECKSUM"
read -p "Please verify the checksum. Press 'y' to continue: " response

if [[ ! $response =~ ^[Yy]$ ]]
then
    echo "Aborting due to user response."
    exit 1
fi
fi

# Warning about formatting and data loss
echo "WARNING: All data on $USB_BLOCK will be lost!"
read -p "Are you sure you want to continue? [y/N]: " confirm
confirm=${confirm:-N}
if [[ $confirm =~ ^[Yy]$ ]]
then
    # 2. Formatting the USB drive
    echo "Formatting USB drive..."
    wipefs -a $USB_BLOCK

    parted $USB_BLOCK mklabel gpt
    parted $USB_BLOCK mkpart BOOT fat32 0% 1GiB
    parted $USB_BLOCK mkpart INSTALL ntfs 1GiB 100%
    parted $USB_BLOCK unit B print

    # Create temporary directories for mounting
    ISO_MOUNT=$(mktemp -d)
    VFAT_MOUNT=$(mktemp -d)
    NTFS_MOUNT=$(mktemp -d)

    # 3. Mounting the ISO
    echo "Mounting ISO..."
    mount $ISO_PATH $ISO_MOUNT

    # 4. Formatting and copying to the USB
    echo "Copying to USB..."
    mkfs.vfat -n BOOT ${USB_BLOCK}1
    mount ${USB_BLOCK}1 $VFAT_MOUNT

    rsync -r --progress --exclude sources --delete-before $ISO_MOUNT/ $VFAT_MOUNT/
```

```
mkdir -p $VFAT_MOUNT/sources
```

```
cp $ISO_MOUNT/sources/boot.wim $VFAT_MOUNT/sources/
```

```
mkfs.ntfs --quick -L INSTALL ${USB_BLOCK}2
```

```
mount ${USB_BLOCK}2 $NTFS_MOUNT
```

```
rsync -r --progress --delete-before $ISO_MOUNT/ $NTFS_MOUNT/
```

```
# 5. Unmounting and power off
```

```
echo "Unmounting drives and syncing data... This may take a while, do not disconnect your USB drive."
```

```
umount $NTFS_MOUNT
```

```
umount $VFAT_MOUNT
```

```
umount $ISO_MOUNT
```

```
sync
```

```
# Remove temporary mount directories
```

```
rmdir $ISO_MOUNT $VFAT_MOUNT $NTFS_MOUNT
```

```
udisksctl power-off -b $USB_BLOCK
```

```
echo "Done! You can now safely disconnect your USB drive."
```

```
else
```

```
echo "Aborted by user."
```

```
exit 1
```

```
fi
```