

Neue Seite

Systemdokumentation: Server Lissabon & Nikosia

Windows Server 2022 • Hyper-V • Sophos NDR (alle VMs produktiv) • Stand:

☐ Serverübersicht

Standort	Servername	Betriebssystem	Virtualisierung	Haupt-VMs
Lissabon	LISSABON	Windows Server 2022 Standard	Hyper-V	NDR-Horst, NDR-Uhura
Nikosia	NIKOSIA	Windows Server 2022 Standard	Hyper-V	NDR-Chekov, NDR-Kirk

⚙ Systemkonfiguration

Beide Server betreiben ausschließlich produktive Sophos NDR-Instanzen unter Hyper-V. Keine zusätzliche Host-Nutzung.

☐ Netzwerkarchitektur

- Ethernet-Ports im **NIC-Team** (Switch Independent, Dynamic Load)
- SFP-Ports als **virtueller Switch** (vSwitch-SFP) für VMs, keine Host-Nutzung
- Klare Trennung zwischen Host- und VM-Netzwerk

☐ Virtuelle Maschinen

Server Lissabon – Sophos NDR-Produktivsysteme

VM-Name	Rolle/Zweck	Netzwerk	Status
NDR-Horst	Sophos NDR – Produktiv	vSwitch-SFP	Produktiv
NDR-Uhura	Sophos NDR – Produktiv	vSwitch-SFP	Produktiv

Server Nikosia – Sophos NDR-Produktivsysteme

VM-Name	Rolle/Zweck	Netzwerk	Status
NDR-Chekov	Sophos NDR – Produktiv	vSwitch-SFP	Produktiv
NDR-Kirk	Sophos NDR – Produktiv	vSwitch-SFP	Produktiv

☐ Sicherheit & Betrieb

- RDP-Port 11334 mit Firewallregel

Alle virtuellen Maschinen sind produktiv und Teil der Sophos NDR-Infrastruktur.

☐ PowerShell-Skript NIC-Team

```
# =====  
# CONFIG  
# =====  
$NicA = "Ethernet" # erste physische NIC
```

```

$NicB      = "Ethernet 2"      # zweite physische NIC
$SwitchName = "vSwitch_Extern" # Name des Hyper-V vSwitch (SET)
$AllowMgmtOS = $true          # Host darf den Switch mitbenutzen

# =====
# HELPER
# =====
function Write-Step($msg){ Write-Host "==> $msg" -ForegroundColor Cyan }
function Try-Stop($script){ try { & $script } catch { } }

# IP-Konfig der aktuellen Management-NIC auslesen (wir nehmen NicA als Quelle)
function Get-SourceIpConfig($ifAlias){
    $ipIf = Get-NetIPInterface -InterfaceAlias $ifAlias -AddressFamily IPv4 -ErrorAction Stop
    $isDhcp = $ipIf.Dhcp -eq 'Enabled'
    if ($isDhcp) {
        return @{ Dhcp=$true }
    } else {
        $ip = Get-NetIPAddress -InterfaceAlias $ifAlias -AddressFamily IPv4 | Where-Object { $_.PrefixOrigin -eq
'Manual' } | Select-Object -First 1
        $gw = Get-NetRoute -InterfaceAlias $ifAlias -DestinationPrefix '0.0.0.0/0' | Sort-Object RouteMetric |
Select-Object -First 1
        $dns = (Get-DnsClientServerAddress -InterfaceAlias $ifAlias -AddressFamily IPv4).ServerAddresses
        return @{
            Dhcp      = $false
            IPAddress = $ip.IPAddress
            PrefixLength = $ip.PrefixLength
            DefaultGateway= $gw.NextHop
            DnsServers = $dns
        }
    }
}

# IP-Konfig auf Zieladapter anwenden
function Apply-IpConfig($ifAlias, $cfg){
    if ($cfg.Dhcp) {
        Write-Step "DHCP auf $ifAlias aktivieren"
        Try { Set-NetIPInterface -InterfaceAlias $ifAlias -Dhcp Enabled -ErrorAction Stop } Catch { }
        Try { ipconfig /renew $ifAlias | Out-Null } Catch { }
    } else {
        Write-Step "Statische IP auf $ifAlias setzen"
    }
}

```

```

# vorhandene IPv4 Adressen entfernen (nur, falls vorhanden)
Get-NetIPAddress -InterfaceAlias $ifAlias -AddressFamily IPv4 -ErrorAction SilentlyContinue | `
    Remove-NetIPAddress -Confirm:$false -ErrorAction SilentlyContinue
# Default Route entfernen
Get-NetRoute -InterfaceAlias $ifAlias -AddressFamily IPv4 -ErrorAction SilentlyContinue | `
    Where-Object {$_.DestinationPrefix -eq '0.0.0.0/0'} | `
    Remove-NetRoute -Confirm:$false -ErrorAction SilentlyContinue
# neue IP & Gateway setzen
New-NetIPAddress -InterfaceAlias $ifAlias `
    -IPAddress $cfg.IPAddress `
    -PrefixLength $cfg.PrefixLength `
    -DefaultGateway $cfg.DefaultGateway -ErrorAction Stop | Out-Null
# DNS setzen
if ($cfg.DnsServers -and $cfg.DnsServers.Count -gt 0) {
    Set-DnsClientServerAddress -InterfaceAlias $ifAlias -ServerAddresses $cfg.DnsServers -ErrorAction Stop
}
}
}

# =====
# 0) Vorprüfung
# =====
Write-Step "Exakte Adapternamen und Status prüfen"
Get-NetAdapter | Select-Object Name, Status, InterfaceDescription | Format-Table

# =====
# 1) Altlasten beseitigen
# =====
Write-Step "Internetverbindungsfreigabe (ICS) deaktivieren (falls aktiv)"
Try-Stop { Stop-Service SharedAccess -Force -ErrorAction SilentlyContinue }
# ggf. bekannte Freigaben deaktivieren (falls cmdlet vorhanden)
Try-Stop { Set-NetConnectionSharing -ConnectionName "Ethernet 2" -SharingMode Disabled -ErrorAction
SilentlyContinue }

Write-Step "LBFO-Teams entfernen (falls vorhanden)"
Try-Stop { Get-NetLbfoTeam | Remove-NetLbfoTeam -Confirm:$false }

Write-Step "Alte vSwitches entfernen, die $NicA / $NicB belegen"
$swToRemove = @()
foreach($sw in (Get-VMSwitch)){

```

```

$desc = $sw.NetAdapterInterfaceDescriptions
if ($desc -and ($desc -match $NicA -or $desc -match $NicB)){
    $swToRemove += $sw.Name
}
}
$swToRemove | Sort-Object -Unique | ForEach-Object {
    Try-Stop { Remove-VMSwitch -Name $_ -Force }
}

# Sicherstellen, dass beide NICs frei und Up sind
Write-Step "Prüfen, ob $NicA / $NicB frei und 'Up' sind"
Get-NetAdapter -Name $NicA, $NicB | Format-Table Name, Status, ifIndex, MacAddress

# =====
# 2) IP-Konfig vom bisherigen Host-Adapter sichern (Quelle = NicA)
# =====
Write-Step "Aktuelle Host-IP-Konfiguration von '$NicA' sichern"
$srcCfg = Get-SourceIpConfig -ifAlias $NicA
$srcCfg

# =====
# 3) SET-vSwitch erstellen
# =====
Write-Step "vSwitch '$SwitchName' (SET) mit $NicA erstellen"
New-VMSwitch -Name $SwitchName `
    -NetAdapterName $NicA `
    -EnableEmbeddedTeaming $true `
    -AllowManagementOS $AllowMgmtOS -ErrorAction Stop | Out-Null

Write-Step "Zweite NIC ($NicB) dem Team hinzufügen"
Add-VMSwitchTeamMember -VMSwitchName $SwitchName -NetAdapterName $NicB -ErrorAction Stop

Write-Step "Load-Balancing = Dynamic setzen"
Set-VMSwitchTeam -VMSwitchName $SwitchName -LoadBalancingAlgorithm Dynamic

# =====
# 4) Host-IP auf vEthernet (Switch) übertragen
# =====
$MgmtIfAlias = "vEthernet ($SwitchName)"
Write-Step "Host-Managementadapter ist '$MgmtIfAlias'"

```

```
# kurze Pause, bis der vEthernet-Adapter erscheint
Start-Sleep -Seconds 3
Apply-IpConfig -ifAlias $MgmtIfAlias -cfg $srcCfg

# =====
# 5) Kontrolle
# =====
Write-Step "Ergebnis prüfen"
Get-VMSwitch -Name $SwitchName | Format-List *
Get-VMSwitchTeam -Name $SwitchName | Format-Table Name, TeamMembers, LoadBalancingAlgorithm
Get-VMNetworkAdapter -ManagementOS | Where-Object {$_.SwitchName -eq $SwitchName} | `
    Format-Table Name, Status, IPAddresses
ipconfig | findstr /I /C:"IPv4" /C:"vEthernet ($SwitchName)"

Write-Step "Fertig. Der Host nutzt jetzt '$SwitchName' (SET) über $NicA + $NicB."
```

© IT-TEAM • Für PDF-Export im Browser drucken → „Als PDF speichern“.

Revision #2

Created 6 October 2025 08:33:42 by Martin Raddei

Updated 6 October 2025 08:57:17 by Martin Raddei