Updating StarWind VSAN Windows Application on a 2-Node Hyper-V Cluster using SCCM

2025

StarWind Documents

Trademarks

Changes

The material in this document is for information only and is subject to change without notice. While reasonable efforts have been made in the preparation of this document to assure its accuracy, StarWind Software assumes no liability resulting from errors or omissions in this document, or from the use of the information contained herein. StarWind Software reserves the right to make changes in the product design without reservation and without notification to its users.

Technical Support and Services

If you have questions about installing or using this software, check this and other documents first - you will find answers to most of your questions on the Technical Papers webpage or in StarWind Forum. If you need further assistance, please contact us .

About StarWind

StarWind is a pioneer in virtualization and a company that participated in the development of this technology from its earliest days. Now the company is among the leading vendors of software and hardware hyper-converged solutions. The company’s core product is the years-proven StarWind Virtual SAN, which allows SMB and ROBO to benefit from cost-efficient hyperconverged IT infrastructure. Having earned a reputation of reliability, StarWind created a hardware product line and is actively tapping into hyperconverged and storage appliances market. In 2016, Gartner named StarWind “Cool Vendor for Compute Platforms” following the success and popularity of StarWind HyperConverged Appliance. StarWind partners with world-known companies: Microsoft, VMware, Veeam, Intel, Dell, Mellanox, Citrix, Western Digital, etc.

Copyright ©2009-2018 StarWind Software Inc.

This article is specifically recommended for multi-location deployments, providing a detailed walkthrough to successfully update StarWind VSAN on a 2-Node Hyper-V Cluster using the SCCM.

Description

In case you have multiple locations with Failover Cluster running StarWind VSAN as a shared storage, installing updates for StarWind VSAN might take a lot of time and effort, thus it makes sense to automate it. This article provides a guide on updating StarWind VSAN using the System Center Configuration Manager (SCCM) package in a 2-node Hyper-V cluster, recommended for cases with distributed multiple locations. **Disclaimer:** StarWind Support does not write scripts and SCCM package on demand. Custom script troubleshooting is not supported. The script provided in this article is as an example and can be customized by the end-user according to his needs. **Prerequisites:** Before proceeding with the update, ensure the following prerequisites are met:

1. **Backup:** Conduct a comprehensive backup of Hyper-V virtual machines and critical data to mitigate potential data loss during the update.
2. **System Center Configuration Manager (SCCM):** Confirm that SCCM is properly configured and operational in the multi-location environment.
3. **StarWind VSAN Update Package:** Download the latest StarWind VSAN update package from the official StarWind website, compatible with the current version.
4. **Cluster Health Check:** Verify the health and status of the 2-Node Hyper-V Cluster across all locations to address any issues before initiating the update.

To avoid production interruption, StarWind VSAN cannot be installed on both cluster nodes simultaneously and must be updated in a one-by-one manner. The update script is designed to minimize the probability of production interruption and is intended to execute the update only on the local node with additional verification steps. Assuming the above, the package has to be deployed to the SCCM collection which consists only of the “first” nodes in the cluster, like SW-HCA-0**1**. Once it is executed successfully (the script execution time could be up to 5 hours) the same package can be deployed to the “second” node in the cluster, like SW-HCA-0**2**. It is recommended to deploy the package to smaller collections and update nodes by clusters to avoid cases when in the same cluster the first node is running on the latest StarWind VSAN build, while the second one is running on the old build for a long time. It’s better to keep the same build for the partner nodes to avoid possible differences that could appear in future builds. **NOTE:** Please keep in mind that it’s mandatory to test the StarWind VSAN update via SCCM on a pilot group before running it for multiple locations. The post-deployment check must be done to confirm that the StarWind VSAN version is updated, StarWind HA devices are synchronized and cluster nodes and roles are up and running. **SCCM Package**

description Before deployment to the selected collection, the SCCM package should be created. The package consists of two files: * starwind-v8.exe – StarWind VSAN installer. The actual version can be downloaded using this link (file name unchanged): https://www.starwindsoftware.com/tmplink/starwind-v8.exe * swupdate.ps1 – PowerShell script which performs preparation steps, StarWind VSAN update, and post-install actions. **Script description** While executing on the StarWind node, the script swupdate.ps1 is writing a log file swupdate.log which is located in the same folder. The log file location can be changed if required, by changing $LogFile variable. While installing, StarWind VSAN writes its own log file vsan_update.log which is also located in the folder with the script. The script swupdate.ps1 performs the following actions:

- check if there is a pending reboot state on the server. There are a lot of registry keys to check it, but most of them were commented on because in most cases StarWind VSAN installation it's not so important. There were only the most important ones uncommented. If a pending reboot is detected, the script will exit with code 3001.
- check nodes state in the cluster – both nodes must be in the online state. If one of the nodes is not in an online state, the script will exit with code 3002.
- check the ISCSI sessions state on the local node. If some session is not connected or not persistent, the script will exit with code 3003.
- kill StarWindManagementConsole process. It's required to run the StarWind VSAN installer. If the process is left running, the script will exit with code 3004.
- update the StarWindX PowerShell module, using the installer file. It has to be installed separately to avoid possible issues later. If the installation is failed, the script will exit with code 3005. To investigate the failure, vsan_update.log can be read additionally.
- check StarWind devices synchronization state on both nodes. If some device is not synchronized, the script will exit with code 3006.
- Identify a partner cluster node and move cluster resources to it. If failed, the script will exit with code 3007.
- pause the local node in the cluster. If that operation is not successful, the script will exit with code 3008.
- update StarWind VSAN on the local node, using the installer file. If the installation is failed, the script will exit with code 3009. To investigate the failure, vsan_update.log can be read additionally.
- check StarWind devices synchronization state on both nodes after the synchronization. If some device is not synchronized, the script will check the synchronization 30 times every 10 minutes until all devices are synchronized. It was implemented to make sure that devices are synchronized after the update in

case full synchronization occurs. If some devices are not synchronized after 5 hours, the script will exit with code 3010. The number of checks and interval can be changed if required, by changing $timeoutSeconds and $timeoutCount variables.

- resume local node in the cluster. If that operation is not successful, the script will exit with code 3011.
- If completed successfully, the script exits with code 0.

Script body:

```
Set-ExecutionPolicy -Scope Process Bypass
$ErrorActionPreference  = "Stop"
$timeoutSeconds         = 600
$timeoutCount           = 30
$LogFile                = "$PSScriptRoot\swupdate.log"
$TimeStamp              = (Get-Date).ToString("yyyy-MM-dd
HH:mm:ss")

Write-Host "$TimeStamp [INFO] Starting update for node
$env:COMPUTERNAME" -ForegroundColor Green
"$TimeStamp [INFO] Starting update for node $env:COMPUTERNAME" |
Out-File $LogFile

function WriteLog{
    [CmdletBinding()]
    param (
        [Parameter(Mandatory=$false, position = 1)]
        [ValidateSet('OK', 'INFO', 'WARNING')]
        $level,
        [Parameter(Mandatory=$false, position = 2)]
        $message,
        [Parameter(Mandatory=$false, position = 3)]
        $exitCode
    )
    try {
        if ($null -eq $level){
            Write-Host "$TimeStamp [ERROR] [$exitCode]
$($error[0].Exception.Message)" -ForegroundColor Red
            "$TimeStamp [ERROR] [$exitCode]
$($error[0].Exception.Message)" | Out-File $LogFile -Append
```

```
            "$TimeStamp #### Exception begin ####" | Out-File
$LogFile -Append
            ">>> Exception Message >>>" | Out-File $LogFile -Append
            $error[0].Exception.Message | Out-File $LogFile -Append
            ">>> Script StackTrace >>>" | Out-File $LogFile -Append
            $_.ScriptStackTrace | Out-File $LogFile -Append
            ">>> Invocation Info >>>" | Out-File $LogFile -Append
            $_.InvocationInfo.line.trim() | Out-File $LogFile -
Append
            "$TimeStamp #### Exception end ####" | Out-File
$LogFile -Append
            exit $exitCode
        }
        else {
            Write-Host "$TimeStamp [$level] $message" -
ForegroundColor Green
            "$TimeStamp [$level] $message" | Out-File $LogFile -
Append
        }
    }
    catch {
        $error[0].Exception.Message | Out-File $LogFile -Append
    }
}

### Check pending reboot requirements

function Test-RegistryKey {
    [OutputType('bool')]
    [CmdletBinding()]
    param
    (
        [Parameter(Mandatory)]
        [ValidateNotNullOrEmpty()]
        [string]$Key
    )

    $ErrorActionPreference = 'Stop'
```

```
    if (Get-Item -Path $Key -ErrorAction Ignore) {
        $true
    }
}

function Test-RegistryValue {
    [OutputType('bool')]
    [CmdletBinding()]
    param
    (
        [Parameter(Mandatory)]
        [ValidateNotNullOrEmpty()]
        [string]$Key,

        [Parameter(Mandatory)]
        [ValidateNotNullOrEmpty()]
        [string]$Value
    )

    $ErrorActionPreference = 'Stop'

    if (Get-ItemProperty -Path $Key -Name $Value -ErrorAction
Ignore) {
        $true
    }
}

function Test-RegistryValueNotNull {
    [OutputType('bool')]
    [CmdletBinding()]
    param
    (
        [Parameter(Mandatory)]
        [ValidateNotNullOrEmpty()]
        [string]$Key,

        [Parameter(Mandatory)]
        [ValidateNotNullOrEmpty()]
        [string]$Value
```

```
    )

    $ErrorActionPreference = 'Stop'

    if (($regVal = Get-ItemProperty -Path $Key -Name $Value -
ErrorAction Ignore) -and $regVal.($Value)) {
        $true
    }
}


$pendingReboot = @(
<# { Test-RegistryKey -Key
'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Component Based
Servicing\RebootPending' } { Test-RegistryKey -Key
'HKLM:\Software\Microsoft\Windows\CurrentVersion\Component Based
Servicing\RebootInProgress' } { Test-RegistryKey -Key
'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto
Update\RebootRequired' } { Test-RegistryKey -Key
'HKLM:\Software\Microsoft\Windows\CurrentVersion\Component Based
Servicing\PackagesPending' } { Test-RegistryKey -Key
'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto
Update\PostRebootReporting' } #>
    { Test-RegistryValueNotNull -Key
'HKLM:\SYSTEM\CurrentControlSet\Control\Session Manager' -Value
'PendingFileRenameOperations' }
    { Test-RegistryValueNotNull -Key
'HKLM:\SYSTEM\CurrentControlSet\Control\Session Manager' -Value
'PendingFileRenameOperations2' }
#   { Test-RegistryKey -Key
'HKLM:\SOFTWARE\Microsoft\ServerManager\CurrentRebootAttemps' }
#   { Test-RegistryValue -Key
'HKLM:\SYSTEM\CurrentControlSet\Services\Netlogon' -Value
'JoinDomain' }
    {
        (Get-ItemProperty -Path
'HKLM:\SYSTEM\CurrentControlSet\Control\ComputerName\ActiveComputer
Name').ComputerName -ne
        (Get-ItemProperty -Path
```

```
'HKLM:\SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName')
.ComputerName
    }
    {
        if (Get-ChildItem -Path
'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Serv
ices\Pending') {
            $true
        }
    }

)

try {
    WriteLog INFO "Looking for pending reboot requirements"
    $needUpdate = 0
    foreach ($element in $pendingReboot) {
        if (& $element) {
            $needUpdate += 1
        }
    }
    if ($needUpdate -ne 0) {
        Write-Error "There is pending reboot requirements detected"
    }
    else {
        WriteLog OK "There are no pending reboot requirements
detected"
    }
}
catch {
    WriteLog -exitCode 3001
}


### Check ClusterNodes
try {
    WriteLog INFO "Get cluster node state"
    foreach ($node in Get-ClusterNode) {
        if ($node.State -eq "Up"){
```

```
            WriteLog OK "Cluster node $node ONLINE"
        }
        else {
            Write-Error -Message "Cluster node $node NOT ONLINE"
        }
    }
}
catch {
    WriteLog -exitCode 3002
}

### Check iSCSI sessions
try{
    WriteLog INFO "Get iSCSI sessions state"
    foreach ($session in Get-IscsiSession) {
        if ($session.IsConnected){
            WriteLog OK "iSCSI session
$($session.SessionIdentifier) is CONNECTED"
        }
        else {
            Write-Error -Message "iSCSI session
$($session.SessionIdentifier) not CONNECTED"
        }
        if ($session.IsPersistent) {
            WriteLog OK "iSCSI session
$($session.SessionIdentifier) is PERSISTENT"
        }
        else {
            Write-Error -Message "iSCSI session
$($session.SessionIdentifier) not PERSISTENT"
        }
    }
}
catch {
    WriteLog -exitCode 3003
}

### Try to kill StarWindManagementConsole process if exist
try {
```

```
    if (Get-Process | Where-Object {$_.Name -eq
"StarWindManagementConsole"}) {
        WriteLog INFO "Try to kill StarWindManagementConsole
process"
        Get-Process | Where-Object {$_.Name -eq
"StarWindManagementConsole"} | Stop-Process
        WriteLog OK "StarWindManagementConsole process killed"
    }
}
catch {
    WriteLog -exitCode 3004
}

### Update StarWindX
try {
    WriteLog INFO "Try to install StarWindX"
    $pinfo = New-Object System.Diagnostics.ProcessStartInfo
    $pinfo.FileName = "$PSScriptRoot\starwind-v8.exe"
    $pinfo.RedirectStandardError = $false
    $pinfo.RedirectStandardOutput = $false
    #$pinfo.CreateNoWindow = $true
    $pinfo.UseShellExecute = $false
    $pinfo.Arguments = "/NORESTART /LOG=vsan_update.log /VERYSILENT
/SUPPRESSMSGBOXES
/COMPONENTS=StarWindxDll,StarWindXDll\powerShellEx"
    $p = New-Object System.Diagnostics.Process
    $p.StartInfo = $pinfo
    $p.Start() | Out-Null
    $p.WaitForExit()
       WriteLog INFO "StarWindX EXECUTED"
    if ($p.ExitCode -ne 0) {
        $swexitcode=$p.ExitCode
        Write-Error "Installation of StarWindX failed. For details
see vsan_update.log in $PSScriptRoot directory. Error code
$swexitcode"
    }
    else {
        WriteLog OK "StarWindX installed successfully"
    }
```

```
}
catch {
    WriteLog -exitCode 3005
}


### Check StarWind HA sync state on local and partner node
try {
    WriteLog INFO "Get StarWind HA devices synchronization status"
    Import-Module StarWindX
    $server = New-SWServer 127.0.0.1 3261 root starwind
    $server.Connect()
    foreach($target in $server.Targets) {
        if ($target.Devices[0].DeviceType -eq "HA Image") {
            if ($target.Devices[0].SyncStatus -eq "1"){
                WriteLog OK "HA device $($target.Alias) on server
$($server.IP) synchronized"
                $partner = New-SWServer
$target.Devices[0].Partners[0].HeartbeatChannels[0].Address 3261
root starwind
                $partner.Connect()
                foreach($partnerTarget in $partner.Targets) {
                    if ($partnerTarget.Devices[0].DeviceType -eq
"HA Image"){
                        if ($partnerTarget.Devices[0].SyncStatus -
ne "1"){
                           Write-Error "Partner HA device
$($partnerTarget.Alias) on server $($partner.IP) not synchronized"
                        }
                        else {
                            WriteLog OK "HA device
$($partnerTarget.Alias) on server $($partner.IP) synchronized"
                        }
                    }
                }
            }
            else {
                Write-Error "Partner HA device $($Target.Alias) on
server $($server.IP) not synchronized"
```

```
            }
        }
    }
    Remove-Module StarWindX
}
catch {
    WriteLog -exitCode 3006
}

### Move cluster resourses to second node
try {
    WriteLog INFO "Get free cluster node"
    $freeClusterNode = Get-ClusterNode | Where-Object {$_.Name -ne
$env:COMPUTERNAME -and $_.State -eq "Up"}[0]
    WriteLog OK "Free cluster node is $freeClusterNode"
    WriteLog INFO "Move cluster resources to $freeClusterNode"
    Get-ClusterNode | Get-ClusterGroup | Move-ClusterGroup -Node
$freeClusterNode | Out-Null
    WriteLog OK "Move cluster resources to $freeClusterNode done"
    WriteLog INFO "Move CSV to $freeClusterNode"
    Get-ClusterSharedVolume | Move-ClusterSharedVolume -Node
$freeClusterNode | Out-Null
    WriteLog OK "Move CSV to $freeClusterNode done"
}
catch {
    WriteLog -exitCode 3007
}

## Suspend cluster node
try {
    WriteLog INFO "Try to PAUSE $env:COMPUTERNAME in cluster"
    Suspend-ClusterNode -Name $env:COMPUTERNAME | Out-Null
    WriteLog OK "$env:COMPUTERNAME PAUSED"
}
catch {
    WriteLog -exitCode 3008
}
```

```
### Update starWind VSAN
try {
    WriteLog INFO "Try to install StarWind VSAN"
    $pinfo = New-Object System.Diagnostics.ProcessStartInfo
    $pinfo.FileName = "$PSScriptRoot\starwind-v8.exe"
    $pinfo.RedirectStandardError = $false
    $pinfo.RedirectStandardOutput = $false
    #$pinfo.CreateNoWindow = $true
    $pinfo.UseShellExecute = $false
    $pinfo.Arguments = "/NORESTART /LOG=vsan_update.log /VERYSILENT
/SUPPRESSMSGBOXES
/COMPONENTS=Service,service\starflb,service\starportdriver"
#/NOCLOSEAPPLICATIONS /NORESTARTAPPLICATIONS
    stop-service -name "starwindservice" -Force
    sleep 300
    get-process | ?{$_.name -eq "StarWindService"} | stop-process -
Force
    sleep 30
    $p = New-Object System.Diagnostics.Process
    $p.StartInfo = $pinfo
    $p.Start() | Out-Null
    $p.WaitForExit()
       WriteLog INFO "StarWind VSAN EXECUTED"
    if ($p.ExitCode -ne 0) {
        $swexitcode=$p.ExitCode
        Write-Error "Installation of StarWind VSAN failed. For
details see vsan_update.log in $PSSCriptRoot directory. Error code
$swexitcode"
    }
    else {
        WriteLog OK "New StarWind VASN build installed successful"
    }
}
catch {
     WriteLog -exitCode 3009
}


### Check SW sync state
```

```
try {
    WriteLog INFO "Get StarWind HA devices synchronization status"
    Import-Module StarWindX
    $syncState = 0
    foreach($element in 1..$timeoutCount){
        $server = New-SWServer 127.0.0.1 3261 root starwind
        if (!$server.Connected) {
            Start-Sleep -Seconds 10
            $server.Connect() | Out-Null
        }
        foreach($target in $server.Targets) {
            if ($target.Devices[0].DeviceType -eq "HA Image") {
                if ($target.Devices[0].SyncStatus -ne "1"){
                    $syncState = 0
                    WriteLog WARNING "HA device $($target.Alias) on
server $($server.IP) not synchronized"
                }
                else {
                    WriteLog OK "HA device $($target.Alias) on
server $($server.IP) synchronized"
                    $syncState = 1
                }
            }
        }
        if ($syncState -ne "1"){
            WriteLog WARNING "NOT all HA device on server
$($server.IP) synchronized. Retry $element of $timeoutCount"
            Start-Sleep -Seconds $timeoutSeconds
        }
        if ($element -eq $timeoutCount) {
            Write-Error "HA devices on the server  $($server.IP)
are not synchronized after $timeoutCount intervals of
$timeoutSeconds seconds"
        }
        else {
            $partner = New-SWServer
$target.Devices[0].Partners[0].HeartbeatChannels[0].Address 3261
root starwind
            $partner.Connect()
```

```
            foreach($partnerTarget in $partner.Targets) {
                if ($partnerTarget.Devices[0].DeviceType -eq "HA
Image"){
                    if ($partnerTarget.Devices[0].SyncStatus -ne
"1"){
                        Write-Error "HA device
$($partnerTarget.Alias) on server $($partner.IP) not synchronized"
                    }
                    else {
                        WriteLog OK "HA device
$($partnerTarget.Alias) on server $($partner.IP) synchronized"
                    }
                }
            }
            WriteLog OK "All HA devices on server $($server.IP) and
server $($partner.IP) are synchronized"
            $server.Disconnect()
            break
        }
    }
    Remove-Module StarWindX
}
catch {
    WriteLog -exitCode 3010
}

## Resume cluster node
try {
    WriteLog INFO "Try to RESUME $env:COMPUTERNAME in cluster"
    Resume-ClusterNode -Name $env:COMPUTERNAME -Failback NoFailback
    WriteLog OK "$env:COMPUTERNAME UP"
}
catch {
    WriteLog -exitCode 3011
}


Write-Host "$TimeStamp [OK] update for node $env:COMPUTERNAME
SUCCESSFUL" -ForegroundColor Green
```

```
"$TimeStamp [OK] update for node $env:COMPUTERNAME SUCCESSFUL" |
Out-File $LogFile -Append
exit 0
```

Contacts

US Headquarters	EMEA and APAC
+1 617 829 44 95	+44 2037 691 857 (United Kingdom)
+1 617 507 58 45	+49 800 100 68 26 (Germany)
+1 866 790 26 46	+34 629 03 07 17 (Spain and Portugal)
	+33 788 60 30 06 (France)

Customer Support Portal: https://www.starwind.com/support

Support Forum: https://www.starwind.com/forums

Sales: sales@starwind.com

General Information: info@starwind.com

StarWind

StarWind Software, Inc. 100 Cummings Center Suite 224-C Beverly MA 01915, USA
www.starwind.com