

Using Emulex HBAnyware Management Utility with PowerShell Scripts



Introduction

Storage administrators can combine powerful tools from Microsoft and Emulex to provide enhanced management and reporting capabilities for Emulex Host Bus Adapters (HBAs) and Converged Network Adapters (CNAs).

The tools are:

- **Microsoft PowerShell®**—PowerShell is a command line shell and scripting language that was initially released as a download add on for Windows Server 2003 and is now inbox with Window Server 2008. PowerShell commands can be used to perform a variety of administrative functions. For example, the following would output information for all processes running on a system:

```
PS C:\Scripts> Get Process
```

Output from a PowerShell script can also be directed to a text file as shown in the following example:

```
PowershellScript >> C:/outputfile.txt
```

- **HBAnyware®**—HBAnyware is the Emulex management utility that provides local and remote management of Emulex adapters with graphical user interface (GUI) and command line interface (CLI) support.

This document provides two examples that demonstrate how HBAnyware commands can be executed in a PowerShell script to provide expanded information on all adapters that can be discovered in a storage area network (SAN).

Both scripts use two HBAnyware CLI commands:

- **ListHBAs**—Provides summary information for all HBAs that can be discovered. The following shows the command and the summary information that would be displayed for each HBA:

```
C:\PROGRA~1\Emulex\Util\HBANYW~1>hbacmd ListHBAs
```

```
Manageable HBA List
```

```
Port WWN : 10:00:00:00:c9:73:67:95
Node WWN : 20:00:00:00:c9:73:67:95
Fabric Name: 00:00:00:00:00:00:00:00
Flags : 8000f100
Host Name : DL585G5-W2K8X32
Mfg : Emulex Corporation
Serial No. : VM80291942
Port Number: n/a
```

- **HBAAttributes**—Displays all attributes for an individual HBA port. The following shows the command using the Worldwide Port Name (WWPN) from the previous example, and the output:

```
C:\PROGRA~1\Emulex\Util\HBANYW~1>hbacmd
HBAAttributes 10:00:00:00:c9:73:67:95
```

```
Host Name : DL585G5-W2K8X32
Manufacturer : Emulex Corporation
Serial Number : VM80291942
Model : LPe12000-M8
Model Desc : Emulex LPe12000-M8 8Gb PCIe Fibre
Channel Adapter
Node WWN : 20 00 00 00 c9 73 67 95
HW Version : 31004549
Opt ROM Version: 5.03A0
FW Version : 1.10A5
Vendor Spec ID : 10DF
Number of Ports: 1
Driver Name : elxstor
Device ID : F100
HBA Type : LPe12000-M8
Operational FW : SLI-3 Overlay
SLI2 FW : 1.10a5
SLI3 FW : 1.10a5
IEEE Address : 00 00 c9 73 67 95
Boot Code : 5.03a0
Driver Version : 5-2.10A7
Kernel Version : 1.10a0
HBA Temperature: Normal
```

Example 1: Save attributes of all HBAs on the SAN to a file

To save the attributes for all adapter ports on a SAN, it is necessary to:

- Execute ListHBAs to get WWPNs for all adapter ports
- Execute HBAAttributes for each port
- Cut and paste output to a master file

For a small SAN with 20 adapter ports, 41 individual steps would be required.

A better option would be running the following PowerShell script with output directed to a text file (comments highlighted in blue):

```
# Execute ListHBAs and store output in $allPorts
$allPorts = .\HbaCmd ListHBAs

# Save lines that include "Port WWN" to $WWNPort
$WWNport = $allPorts -match "Port WWN"

# Remove "Port WWN : " text and save in $WWNentry
$WWNentry = $WWNport -replace "Port WWN : ", ""

# Execute HBAAttributes for each WWN value in
$WWNentry
$n = $WWNentry.count;
while ($n -gt 0) {.\hbacmd HBAAttributes
$WWNentry[$n-1];$n=$n-1}
```

Example 2: Save attributes of all HBAs on the SAN

This example is a more advanced script that also saves the attributes for all adapter ports. The difference is summary information and port attributes are saved in objects, which could be building blocks for scripts that perform other tasks.

Two functions are defined and executed in the script:

- **Get Ports**—Executes ListHBAs and saves the summary information for each port
- **Main**—Runs Get Ports and then executes HBAAttributes for each port

The final step of the script calls the Main function.

```
# Get-Ports function:
# Discover adapters and return object array with
summary information for each.
# Fields: PortWWN, NodeWWN, FabricName,
Flags, HostName, Mfg, SerialNumber

function Get-Ports {
    # Declare AllPorts array
    $AllPorts = @();

    # Execute ListHBAs and store response in $rawStrings
    $rawStrings = ./hbacmd ListHBAs

    # Create PortObject for each adapter port
    $index = 0
    foreach ($line in $rawStrings) {
        if ($line -match "Port WWN") {

            # Create PortObject class/object with fields for
            summary information.
            $PortObj = "" | Select PortWWN, NodeWWN,
            FabricName, Flags, HostName, Mfg, SerialNumber

            $PortObj.PortWWN = $line.TrimStart("Port WWN :");
            $AllPorts += $PortObj;
            $index++;
        }

        if (($line -match "Manageable HBA List") -or
        ($line -match "")){
        }
        else {
            if ($index -le 0) {
                break
            }
            else {

                # Store the summary information into an index
                array that could
                # be referenced by others commands that require
                these values.
                # Label headers at each line are trimmed, only
                values are stored.
                if ($line -match "Node WWN") {
                    $AllPorts[$index - 1].NodeWWN = $line.
                    TrimStart("Node WWN :")
                }

                if ($line -match "Fabric Name") {
                    $AllPorts[$index - 1].FabricName = $line.
                    TrimStart("Fabric Name:")
                }
            }
        }
    }
}
```

```
if ($line -match "Flags") {
    $AllPorts[$index - 1].Flags = $line.
    TrimStart("Flags :")
}

if ($line -match "Host Name") {
    $AllPorts[$index - 1].HostName = $line.
    TrimStart("Host Name :")
}

if ($line -match "Mfg") {
    $AllPorts[$index - 1].Mfg = $line.
    TrimStart("Mfg :")
}

if ($line -match "Serial No.") {
    $AllPorts[$index - 1].SerialNumber = $line.
    TrimStart("Serial No. :")
}
}
}
}

return $AllPorts
}

# Main function :
# Call Get-Ports to acquire indexed PortWWN values and
# run HBAAttributes for all ports.

function Main {
    $Ports = Get-Ports;
    $Ports.Count;

    if ($Ports.Count -ne 0) {
        foreach ($objPort in $Ports) {
            ./hbacmd hbaattributes $objPort.PortWWN
        }
    }
}

# Call Main function.
#
Main;
```

Conclusion

PowerShell significantly enhances the capabilities of the HBAAnyware CLI. With PowerShell scripting, IT administrators can quickly create custom tools to report and manage the use of Emulex adapters throughout their SAN.

References:

www.microsoft.com/technet/scriptcenter/topics/winpsch/manual/default.msp
www.emulex.com/products/management-software/hbanyware/list/overview.html

Resources:

www.emulex.com/solutions/windows_server_2008/index.jsp