Is this thing on? Using the PowerShell Test-Connection CmdLet

As a lover of all things PowerShell, I'm constantly doing queries and tasks against large pools of computer objects from my Active Directory network. One of the continuing challenges I run into is machines which are registered computer objects domain, but not online. It's not a problem that they aren't online in general, but the issue is that every remote task I try to run, the script throws an error and takes time to timeout any tasks before moving on.

This kind of error result can either confuse the overall script that I'm trying to run, or it can trigger the failure and exit which really leaves us in a rough spot.

The Test-Connection CmdLet

There is a nifty little PowerShell CmdLet called **Test-Connection** which will be our friend in these cases. Effectively, this is running a ping test against the server. By using ICMP, we confirm if the target system is responding, but rather than running a ping command externally and having to parse the output for result codes, we get to use this really cool native PowerShell CmdLet.

There are quite a few parameters, but the ones that matter to us will be the –**Computername, -BufferSize, -Count,** and **-quiet**. Here is what those all mean to us:

- **-Computername**: Fairly self-explanatory
- **-BufferSize**: the size of the ICMP packet which we keep small for efficiency
- **-Count**: how many ICMP packets to send
- **-Quiet**: reduce the output because we don't need anything sent to the screen

So what we will want to do for the quick, but effective test of a system's online status is to send one 16 byte packet with no output. Normally we can run the output which shows us the results of the ping including the source system, destination system, and the IP addresses (IPv4 and IPv6).

The output comes in table format normally which is handy for many other tasks and reports, but we simply need a quick check to confirm if we can take additional actions against a network resource. If we add each of our parameters in steps, we can see how the results change.

The command we will get to in the end is as follows:

```
Test-Connection -Computername $client -BufferSize 16 -Count 1 -Quiet
```

As you can see, the full command gives us only a Boolean output of either True or False. We can use this easily with an If statement

The overall process – Query AD and check if a machine is online

My use case that I'm illustrating is quick and easy. I want to run a script block against all of my computers in Active Directory running Windows 7. For the purposes of our example we will be writing a simple piece of screen output stating the computer name and that it is online.

Because we are using Active Directory as the source for computer objects, we need to load the Active Directory PowerShell module first:

```
Import-Module ActiveDirectory
```

Now we create a collection using a simple Get-ADComputer query and our condition in this case is we are only going to get the computers with Windows 7 as the operating system:

```
$computers = Get-ADComputer -Filter {OperatingSystem -like "Windows 7*"}
```

Next we will loop through the collection and assign a variable named $client using the computer name field:

```
ForEach ($computer in $computers) {

$client = $Computer.Name
```

Now the fun stuff. Normally we would just hammer ahead with the script actions we want to run against the computer, but instead we run the **Test-Connection** CmdLet inside an If statement so that only a result of True will initiate our script block inside the curly braces of the If statement logic:

```
if (Test-Connection -Computername $client -BufferSize 16 -Count 1 -Quiet)
{
    Write-Host $client is online
    }
}
```

And we are done! It really is that simple. Just replace the Write-Host line with whatever you wish to do on your target machine and you well ensure that it only runs if the machine is online.

Simple example, powerful tool

While we have chosen a very simple example here, the idea is that you can take the same process, apply more content into the script block inside your If statement. In addition, we now have a powerful and effective way to programmatically act upon multiple systems more quickly, and by adding a simple command block we know that the systems will be online and not cause our overall script to halt from a timeout or general error.

Here is the full script from our example:

```
Import-Module ActiveDirectory

$computers = Get-ADComputer -Filter {OperatingSystem -like "Windows 7*"}
 ForEach ($computer in $computers) {
```

```
$client = $Computer.Name

if (Test-Connection -Computername $client -BufferSize 16 -Count 1 -Quiet)
{
    Write-Host $client is online
    }
}
```

For a the full Microsoft TechNet article on the **Test-Connection** CmdLet, you can go here: http://technet.microsoft.com/en-us/library/hh849808.aspx

Hope this is helpful!