Regular expressions in PowerShell

At the moment I am working to deploy a new product into our environment at work, and like all new products there are snags and bugs that need to be worked out. As part of the deployment I am required to provide a collector with a bunch of server names so that it can go off and do is job, easy enough, or so I thought. After a bit of playing around we noticed that one of the even logs was winging about some dead computers (list was generated from AD) and I needed to remove them from the agents list in order for the agent to run 100% properly.

I started doing this manually and after about the 10th computer I got annoyed and decided to do this using PowerShell.

The error line in the log always contains text similar to this “Could not connect to computer [xxx]”. So off the bat I know that my regular expression should look something like this “Could not connect to computer \[(.*?)\]”.Here is a nice online tool for testing expressions that I like to use, and here is a good site to learn a bit more about regular expressions.

If there are any lines that match this expression I will want to remove them from the config file programmatically (not covered in the sample script). So after a bit of bashing around on the keyboard I was able to produce the following PowerShell script.

```
# Demo Regular Expression app
Clear-Host

$inputFile = "c:\log.log";
$inputData = Get-Content -Path $inputFile;
$pattern = "Could not connect to computer \[(.*?)\]";
$matches = [regex]::Matches($inputData, $pattern);

if( $matches.Count -gt 0 )
{
  foreach( $match in $matches )
  {
          $computer = $match.Groups[1].Value;
          # Add logic here for whatever
  }
}
```

That’s all there is to working with regular expressions in PowerShell, feel free to stop reading here or if you up to it read on for a line by line breakdown of the script.

OK, so let’s begin with the breakdown of the script.

```
Clear-Host
```

Hopefully self explanatory, this will clear all output that is in the current command window.

```
$inputFile = "c:\log.log";
```

Here I am defining the input file that I am going to be running my expressions against; this can be any file URI.

```
$inputData = Get-Content -Path $inputFile;
```

I am storing the contents of the file into a variable named $inputData – this is not the best approach for this especially when dealing with large files. To work with big input files I would recommend reading the file in clunks, but for my example and use case this is fine (the log files roll at 10Mb).

```
$pattern = "Could not connect to computer \[(.*?)\]";
```

This is the pattern that I am going to be running against the $inputData to find all the computer names in the log file. I posted some link above that can point you in the right direction if you are not 100% sure on how to use regular expressions.

```
$matches = [regex]::Matches($inputData, $pattern);
```

Here I am making use of the Regex .net class and calling the Matches function to find all the occurrences of the given regular expression for the given text.

```
if( $matches.Count -gt 0 )
```

The next thing to do is to ensure that we have some matches before continuing on with the script. I make use of the Count property of the $matches object to check that we have 1 or more matches before continuing.

```
foreach( $match in $matches )
```

It is time to loop through the matches and do something with them, by machining use of the foreach loop in PowerShell I am able to address every Match object individually in my $matches collection.

```
$computer = $match.Groups[1].Value;
```

The last thing that I need to do is get the value for the current match (the computer name). This is done by accessing the Groups for the current match and selecting the group that contains the value that you want. In this case the group that I want is the only group in my expression (.*?) so I can happily access it by using Groups[1]. If the match I wanted was say the 7'th group in my expression I would access it by using Groups[7] and so on and so forth.

And that's all there is to using regular expressions in PowerShell.

If you have any questions feel free to post them