PowerShell cURL – Yes It Exists – Invoke-RestMethod

cURL is a tool we all find very useful, and if we are PowerShell users, we often want to have the combination of PowerShell and cURL. With PowerShell 3.0, one of the really great CmdLets that is available is **Invoke-RestMethod**. This handy little CmdLet allows us to now use the PowerShell scripting language to access HTTP resources using the native HTTP methods (GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, OPTIONS, MERGE, PATCH).

If you are familiar with curl, you must be thinking the same thing as I am: this sounds exactly like cURL in PowerShell!

Now, let's take a look at a step by step guide to get using PowerShell cURL!

The Same Thing as PowerShell cURL!

If you have needed to access HTTP resources in the past, you would usually use a command line tool like **cURL** to be able to interact with HTTP resources through batch processes and scripts. This works great of course, but not if you are spending most of your time in PowerShell. Now now that we have the ability to use the core **Invoke-RestMethod** CmdLet, let us quickly review how it works (which is exactly like cURL)

How the cURL PowerShell Equivalent Works

Let's pretend we need to consume some XML data and render it to a file on our Windows server. The way that we would have done this before with curl would be using the following command line:

```
curl.exe www.discoposse.com/index.php/feed > C:TempDiscoPosseFeed.xml
```

Now that was not a difficult thing to deal with, but the point of our exercise is to eliminate unnecessary third party tools and functions where there are native PowerShell CmdLets available. This can be very hand when we want the functionality of curl in a PowerShell script.

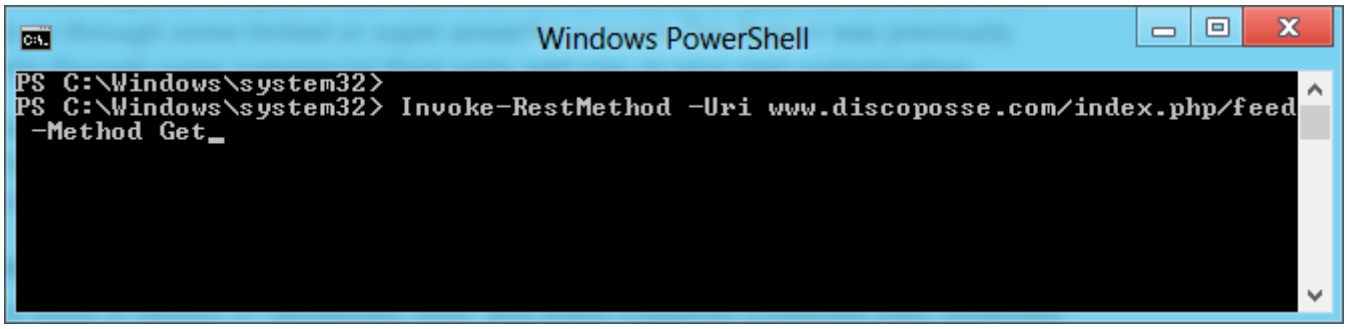
This is where our newly available **Invoke-RestMethod** comes in. By leveraging this CmdLet we no longer need to launch the **cURL** utility. We can get the same functionality in PowerShell just like curl.

PowerShell cURL Like Command – Example

Now, we will try the new way of doing this.

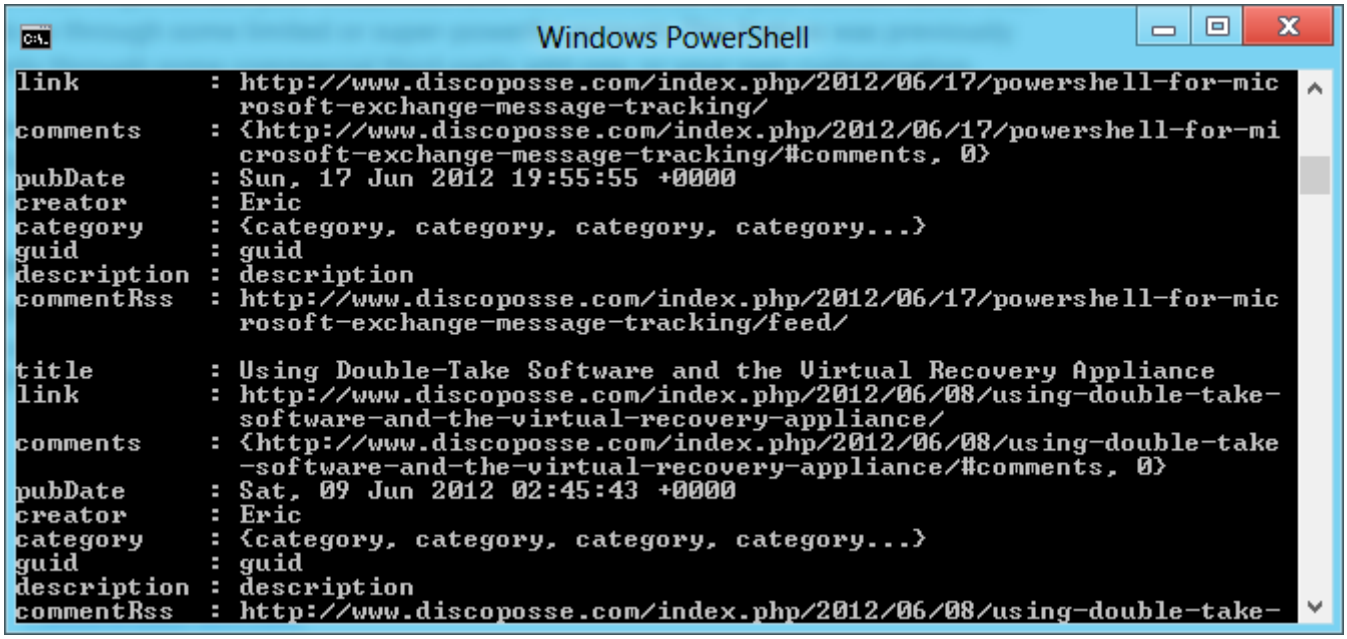
Our new command we will use to begin is this:

```
Invoke-RestMethod -Uri www.discoposse.com/index.php/feed -Method Get
```



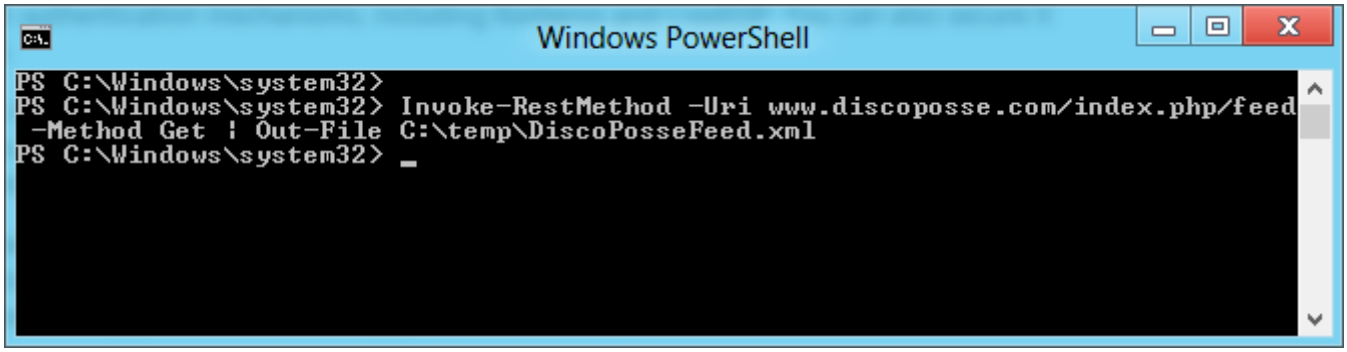
Simple enough to begin with. This uses the URI of my news feed and runs the GET method to retrieve it. The default method for the CmdLet is GET so you could choose to leave out the ***-Method Get*** parameter.

Now we launch the PowerShell command and see that it writes the output to screen. As it should of course because we haven't done anything to redirect the output to file yet.

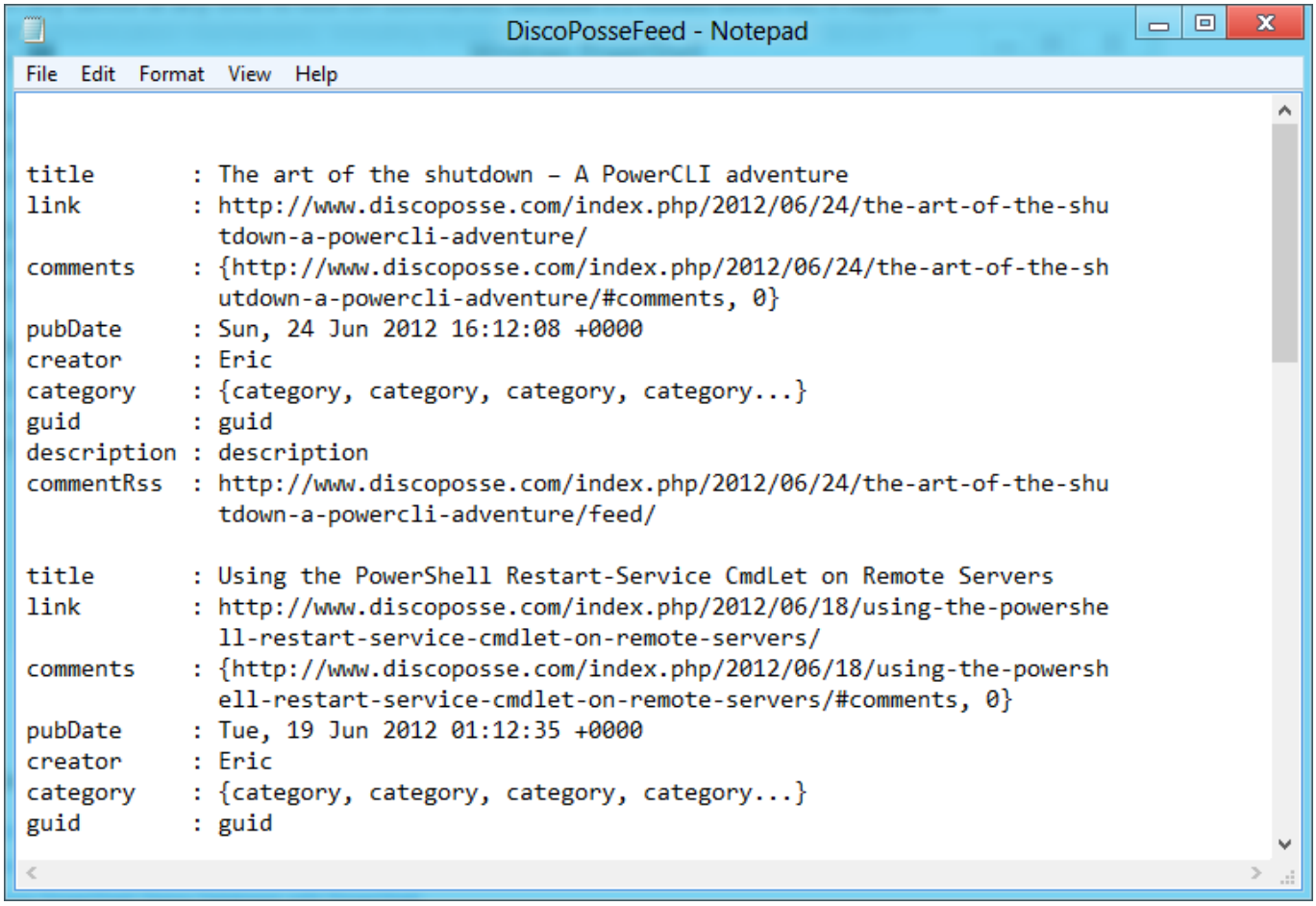


So our next step we do is our usual way of using the pipeline to output to the file system to an XML file.

```
Invoke-RestMethod -Uri www.discoposse.com/index.php/feed -Method Get | Out-File C:TempDiscoPosseFeed.xml
```



Now let's take a look at the file itself to see what we have gotten from the process so far.

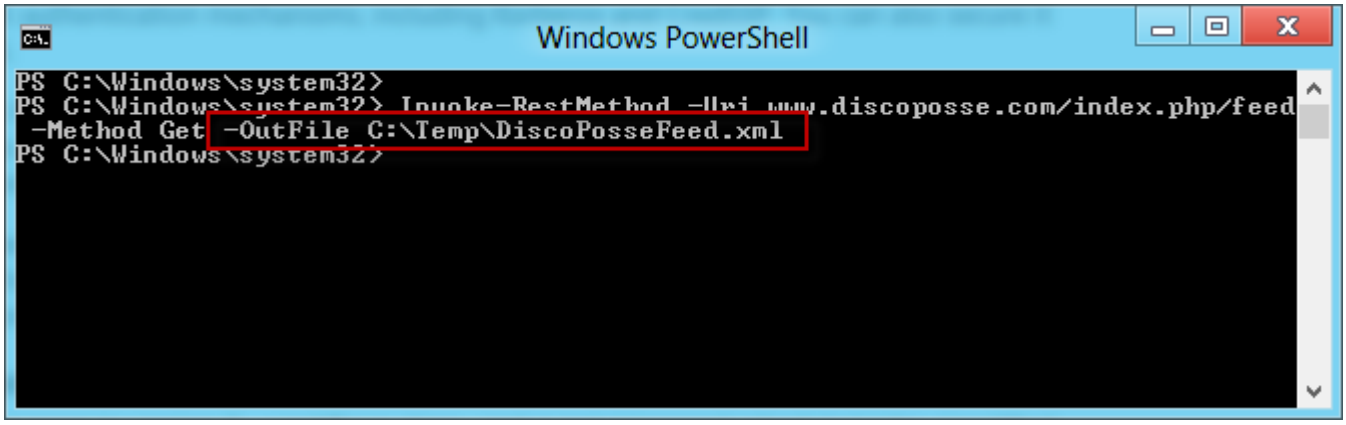


It seems that we don't quite have what we want here doesn't it? We want the file to be in XML format. After all, the content being read from the HTTP resource is XML output. If you look back at the CmdLet output to screen you will see that it looks the same as our file.

This is our eureka moment! We have somehow rendered the XML into more human readable output. This is a cute little trick, but not what we were after. This is a new "feature" of the **Invoke-RestMethod** CmdLet that it renders XML output as a list.

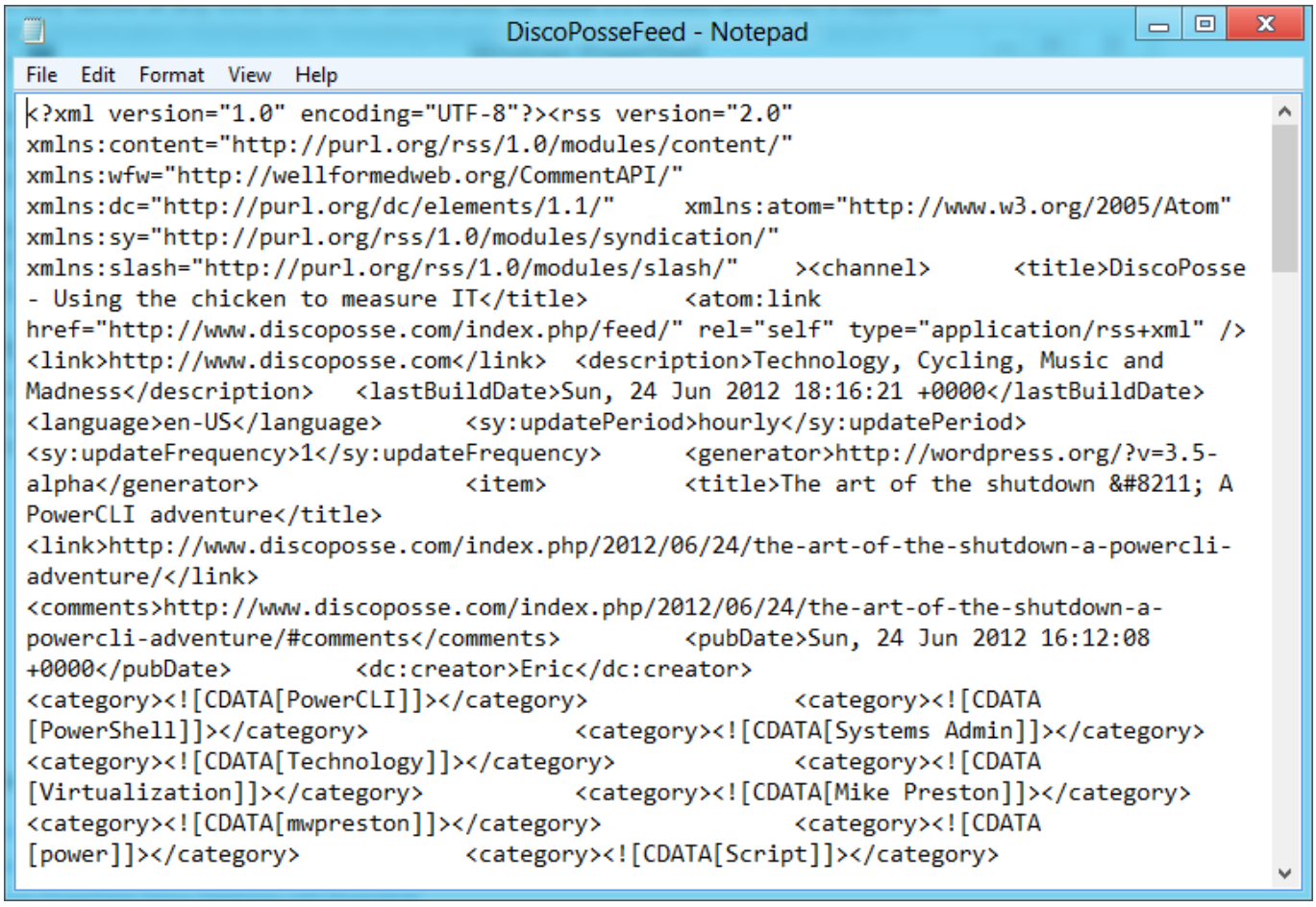
The fix for this situation is simple. Rather than using the traditional method of the pipeline to an **Out-File** CmdLet, we just have to add the **-OutFile** parameter to the command and this takes care of the output to the file system for us.

> Invoke-RestMethod -Uri www.discoposse.com/index.php/feed -Method Get -OutFile C:TempDiscoPosseFeed.xml



Now we open the output file and we will see that it is in the exact format that we would expect an XML stream to be. Much better!

The Power of cURL in PowerShell



Summary: PowerShell cURL – How to Get It!

To review what we have done here, we have used the **Invoke-RestMethod** CmdLet to perform the Get HTTP method against an available URL and rendered the output to a file. In this case it was an XML file that we have created. We have used PowerShell to do exactly what we would normally do with cURL.

The only downside to this is that it is not yet available because it is only a part of the PowerShell 3.0 environment. Never fear, we aren't far away from having this in general availability. Then we can leverage this and many of the othe exciting features of PowerShell 3.0 and all it has to offer.

If you are a PowerShell and cURL user, be sure to check out Downloading Text and Binary Objects with cURL.