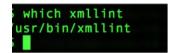
## **Experimenting with the REST API – What Tools can I use? – Part IV**

In part III of this blog series on the common REST API tools, we started looking at some of the more commonly used tools that can be used to interact with the REST API. We very briefly mentioned one such utility, that being xmllint(1) which is command-line utility that is primarily used to parse XML files and identify any errors in your code (which is such a common occurrence when working with XML) before you try to use it with the NSX for vSphere Manager. If you are uncertain if the XML payload is correct, then you really are going to have a difficult time working with the REST API and NSX for vSphere Manager.

However, there is another very nice use of xmllint(1) and this is what I am going to focus on in this blog post.

Recall in part III we mentioned there is an issue currently with the NSX for vSphere platform, and that is that JSON is **NOT** currently **SUPPORTED**. So what do we want to use xmllint(1) for then? OK, let's start at the beginning, and all will become clear.

On Mac OS X xmllint(1) is located in /usr/bin/xmllint



And for the Man Pages:

```
$ man -k xmllint
xmllint(1) - command line XML tool
$ man xmllint
```

So, let us take the following example where we want to issue a very basic REST API request to the NSX for vSphere Manager, but in this case we want to return the data in JSON. I always like to use the following as an example:

1. The following REST API Request return's the NSX for vSphere Manager API versions in XML:

https://wordhtml.com/

2. The following REST API Request return's the NSX for vSphere Manager API versions in JSON:

But wait! As you can see even just trying to return one of the most basic REST API Requests for the NSX for vSphere API versions has caused an error where we are not able to return the REST API Request in JSON. What has really occurred here is that the JSON object could not be decoded.

So, what does this all mean then? Not clear yet? Well, in the case we are now forced to use the payload returned in XML! As you can see in the above example, that is a very basic REST API Request and see what has been returned! Easy to read? Indeed it is not!

Enter xmllint(1)! Here we can now use xmllint(1) to make this readable.

3. This time when we issue the REST API Request we are going to re-direct the output as in the following example:

```
curl -k -u <username>:<password> -H "Accept:application/xml" -H "Content-Type:application/xml" -X GET https://<NSX Manager IP Address>/apiersions > /var/tmp/apiversions.xml
```

Here you can see we have redirected this REST API Request to a file "apiversions.xml" in the directory /var/tmp.

4. Now we can finally add the magic of xmllint(1) to make the contents of this file humanly readable. We do this by issuing the following command:

## \$ xmllint --format /var/tmp/apiversions.xml

Now instead of the trying to read the XML data as depicted in step 1, you can now see this is much more readable (this output has been truncated, as this is just an example):

This was just one example of using the built-in MAC OS X operating system tools, curl(1) and xmllint(1) to work and interact with the REST API and NSX for vSphere Manager to make the reading of the output much easier and less prone to error.

https://wordhtml.com/