**How To Query and Parse a REST API with PowerShell**

*Adam takes a close look at the Invoke-WebRequest and Invoke-RestMethod commands.*

* **By Adam Bertram**
* **04/02/2019**



APIs seem to be everywhere these days. Every popular online service has one.

APIs allow developers to interact with online services without actually having to point and click their way through a UI. Instead, many services offer APIs -- in particular, REST APIs -- that offer up an industry-standard way of exposing information to the world.

When using PowerShell to work with REST APIs, you'll eventually come across two commands: Invoke-WebRequest and Invoke-RestMethod. Both of these commands send and receive HTTP data to/from various HTTP endpoints and can be used to interact with REST APIs. In a nutshell, the Invoke-RestMethod command is everything Invoke-WebRequest is, but with more built-in JSON parsing.

In this article, we'll cover both of these PowerShell commands and learn how each differs and, ultimately, which command you should use to work with REST APIs.

Let's start with a simple example. We'll need an existing REST API to work with. Browsing around on the Internet, I come across a REST API called [**Cat Facts**](https://alexwohlbruck.github.io/cat-facts/docs/). This API returns various facts about our feline friends and is a great API to demonstrate how Invoke-RestMethod works.

Every REST API has a base URI and an endpoint. Our Cat Facts API has a base URI of [**https://cat-fact.herokuapp.com**](https://cat-fact.herokuapp.com/) and has an endpoint of /facts, making the endpoint URI [**https://cat-fact.herokuapp.com/facts**](https://cat-fact.herokuapp.com/facts).

Invoke-RestMethod differs from Invoke-WebRequest by its passing ability. Invoke-RestMethod natively understands the output a REST API method returns, which is typically JSON. When the API does return JSON, Invoke-RestMethod will parse the JSON and return useful PowerShell objects. To demonstrate, I'll use the Invoke-RestMethod command's brother, Invoke-WebRequest. The Invoke-WebRequest command performs a similar function by sending HTTP verbs to Web services but does not have the parsing ability that Invoke-RestMethod does.

If I run Invoke-WebRequest against our Cat Facts API, we have to look at the Content property, which contains a whole lot of unparsed JSON:

PS C:\> (Invoke-WebRequest -Uri $endpointUri).Content

{"all":[{"\_id":"5894af975cdc7400113ef7f9","text":"The technical term for a cat's hairball is a bezoar.","user":{"\_id":"587288f6e6f85e64ae1c7ef7","name":{"first":"Alex","last":"Wohlbruck"}},"upvotes":[{"user":"5872812829f7f43cacb0c4de"}]},{"\_id":"58e007db0aac31001185ecf7","text":"There are cats who have survived falls from over 32 stories (320 meters) onto concrete.","user":{"\_id":"58e007480aac31001185ecef","name":{"first":"Kasimir","last":"Schulz"}},"upvotes":[{"user":"5872812829f7f43cacb0c4de"}]},{"\_id":"58e008780aac31001185ed05","text":"Owning a cat can reduce the risk of stroke and heart attack by a third.","user":{"\_id":"58e007480aac31001185ecef","name":{"first":"Kasimir","last":"Schulz"}},"upvotes":[{"user":"5872812829f7f43cacb0c4de"},{"user":"5c2dd8241d3f5500149d59fc"}]},{"\_id":"58e008800aac31001185ed07","text":"Wikipedia has a recording of a cat meowing, because why not?","user":{"\_id":"58e007480aac31001185ecef","name":{"first":"Kasimir","last":"Schulz"}},"upvotes":[{"user":"5872812829f7f43cacb0c4de"}]},{"\_id":"58e008b80aac31001185ed0d","text":"Adu

<SNIP>

I could read and parse all of this JSON myself with a command like ConvertFrom-Json, but why would I when I could just use Invoke-RestMethod?

If I query this same API with Invoke-RestMethod, I get a familiar PowerShell object back:

PS C:\> Invoke-restmethod -Uri $endpointUri

all

---

{@{\_id=5894af975cdc7400113ef7f9; text=The technical term for a cat's hairball is a bezoar.; user=; upvotes=System.Object[]}, @{\_id=58e007db0aac310011...

We didn't even need to specify the Content property at all. Invoke-RestMethod understood that we were just looking for the HTTP content, and displayed and parsed it for us.

The Invoke-RestMethod also has some other useful features when working with REST APIs such as authentication. Some APIs require you to authenticate via a username/password with basic authentication or perhaps an OAuth token. We can use the Credential parameter to pass a username and password if the API supports basic authentication. Usually, though, you'll find a lot of APIs require an OAuth token. For a great explanation on working with OAuth with PowerShell, check out [**Stephen Owen's blog post**](https://foxdeploy.com/2015/11/02/using-powershell-and-oauth/).

The Invoke-RestMethod command allows you to pass OAuth tokens and other information the API needs via HTTP headers using the Headers parameter. Perhaps the REST API is set up to accept OAuth tokens using the command Authorization key. We can pass our OAuth token with Invoke-RestMethod like so:

PS> Invoke-RestMethod -Uri 'https://cat-fact.herokuapp.com/facts' -Headers @{ 'Authentication' = 'Bearer xxxxxxxxxxxxxxxx' }

This command isn't just relegated to retrieving information via APIs using the HTTP GET method, which is sent by default. We can use many different HTTP methods like POST, PATCH, PUT and UPDATE, as well. We simply need to specify the method via the Method parameter and then provide a HTTP body using the Body parameter.

Perhaps we have a REST API that accepts a POST request and expects the body to be in JSON format. We've already got our JSON payload saved in a variable called $jsonSample that looks something like:

{

"SomeElement" : "Foo"

}

We can pass this JSON payload to the Body parameter and optionally use the ContentType parameter to tell the API that the payload is JSON. This step isn't always required but it is common. Our Invoke-RestMethod API call would then look something like this:

PS> $token = 'xxxxxxxxxx'

PS> $params = @{

Uri = 'https://cat-fact.herokuapp.com/facts'

Headers = @{ 'Authorization' = "Bearer $token" }

Method = 'POST'

Body = $jsonSample

ContentType = 'application/json'

}

PS> Invoke-RestMethod @params

The command above will authenticate to the URI endpoint, pass the JSON sample via the HTTP POST method and tell the API that the content is JSON. If the API returns a response, Invoke-RestMethod will return it to the console already parsed.

The Invoke-RestMethod command is a great way to interact with REST APIs in PowerShell. Although most -- if not all -- of its functionality can be performed via a combination of Invoke-WebRequest, ConvertTo-Json and ConvertFrom-Json commands, it's a great all-in-one command that does all of the heavy lifting for you.