Extracting API Data Using PowerShell and Loading into SQL Server

By: Tim Smith (/sqlserverauthor/92/tim-smith/) | Updated: 2015-02-03 | Comments (4) | Related: More (/sql-server-dba-resources/) > PowerShell (/sql-server-tip-category/81/powershell/)

Problem

We'd like to minimize using third party tools to extract data from APIs to load into SQL Server and wanted to know if we could implement solutions with PowerShell without too much overhead.

Solution

Yes, we can extract API data using PowerShell similar to how we can extract the same data in C#. Generally, we will get the data in XML or JSON (in some cases, we'll directly parse HTML) and add the data into SQL Server. We can also bypass some tools that may add additional overhead (or loading) to get these data. The below is a guide to extracting data from an API, in this case using FRED's (http://api.stlouisfed.org/docs/fred/fred.html) West Texas Intermediate and provides an overall guideline of how to do so with other APIs.

Always Read the Documentation

Always read the API documentation because it will tell you what data are provided, how to access the data, what type of format it will provide, and will sometimes provide an example. In the case of FRED (http://api.stlouisfed.org/docs/fred/fred.html), we can read its documentation (http://api.stlouisfed.org/docs/fred/) to get a feel for what we'll be extracting, specifically how the data and topics are structured. Also, if the API is not well known, this documentation will also provide us with ideas about how to validate before we add the data - we don't want to add *anything* into our database. Finally, many APIs, like FRED, will require an API key, and developers often need to register for a key in order to proceed. This example does not provide a key, so if you want to emulate the example, you can register at FRED to get your free API key.

Converting Formats

After reading the documentation, we know that we want the West Texas Intermediate in the JSON format. In order to read JSON into SQL Server, we will need to convert the JSON syntax into a TSQL syntax (some developers may convert the JSON objects into a data table and perform a SQL Bulk Copy); in this case, I use a T-SQL syntax for easy debugging, but other methods are available. Also, keep in mind that if you're using a document oriented database, like MongoDB, you can perform a direct insert and bypass converting data. Below is a simple example of a JSON document that we convert from and store in the object **$converted**; note how the fields become the object's properties:

```
$json = '{"id": 1, "field": "This is a JSON document", "date": "2014-06-04"}'
$converted = ConvertFrom-Json $json
Write-Host "Our ID is" $converted.id
Write-Host "Our Field is" $converted.field
Write-Host "Our Date is" $converted.date
```

that when we pull data from an API, we must convert the data to the appropriate format, so if it's XML, JSON, or HTML, left these, unless we intend to store the data directly in the database (I would add a data validation method before this in that case).

```
Our ID is 1
Our Field is This is a JSON document
Our Date is 2014-06-04
```

Data Recon

We now know what we want to get and how to convert the data; now we need to build our output. First, we want the dates and values of West Texas Intermediate, so we'll build a table with two columns:

```
CREATE TABLE tb_FRED_WTI(
 WTIDate DATE,
 WTIValue DECIMAL(22,4)
)
```

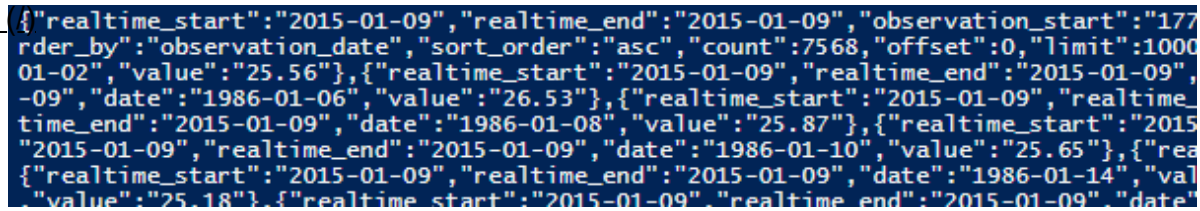
Also, we need a quick PowerShell function that will execute T-SQL and the below script does that (we provide the server and database name and the T-SQL text to execute):

```
Function Add-APIData ($server, $database, $text)
{
    $scon = New-Object System.Data.SqlClient.SqlConnection
    $scon.ConnectionString = "SERVER=$server;DATABASE=$database;Integrated Security=true"

    $cmd = New-Object System.Data.SqlClient.SqlCommand
    $cmd.Connection = $scon
    $cmd.CommandText = $text
    $cmd.CommandTimeout = 0

    $scon.Open()
    $cmd.ExecuteNonQuery()
    $scon.Close()
    $cmd.Dispose()
    $scon.Dispose()
}
```

Now, we need to extract the JSON string. Microsoft provides us with a useful class, the Web Client class, for working with data. It comes with a method called *DownloadString* and this method can store a string in an object. Like creating new connections and commands, we'll create a new object that will allow us to use this class:

```
###  This example does not provide an API key; you can get a free API key from FRED, then input it in the below obje
$api_key = ""
[string]$webstring = "http://api.stlouisfed.org/fred/series/observations?series_id=DCOILWTICO&api_key=$apikey&file_t
$webget = New-Object System.Net.WebClient
$result = $webget.DownloadString($webstring)
```

The **$webget** becomes our object where we can call the method, *DownloadString*, to get the data from an API call (in this case, the API call string is **$webstring** and note that the API key (stored in **$apikey**) is not provided because you'll need to register to get one with FRED. Once we do this, and **Write-Host $result** we'll see a large JSON document that we need to convert:




```
$result = ConvertFrom-Json $result
$add = @()
```

The above converts the JSON and stores it back in the **$result** object; also, I've set up an empty array which will store the insert statements eventually. Once we convert **$result** from JSON, we'll see (when we print **$result**) that the date we're seeking is stored in the sub-document "observations"; this means that in each value of **$result**.*observations* we'll find the date and value fields that we want to store in our database. That means that we need each date and each value of **$result**.*observations*, so we'll perform our loop:

```
foreach ($r in $result.observations)
{
        $add += "INSERT INTO tb_FRED_WTI VALUES ('" + $r.date + "','" + $r.value + "')" + $nl
}
```

This looks good, but we have a problem: we haven't validated our data yet, and if we look at our data, we'll notice that on Christmas day of 2014 the value is "." because West Texas Intermediate wasn't reported. We want to take the time to validate our data here; while FRED is trusted and used very frequently, I still like to think about data validation, such as max lengths, invalid values, etc. For this article, I'll only validate the invalid values:

```
if ($r.value -ne ".")
{
 $add += "INSERT INTO tb_FRED_WTI VALUES ('" + $r.date + "','" + $r.value + "')" + $nl
}
```

Now, outside the loop, we add the data to our database:

```
Add-APIData -server "TIMOTHY\SQLEXPRESS" -database "MSSQLTips" -text $add
```

And review the results ...

```
SELECT *
FROM MSSQLTips.dbo.tb_FRED_WTI
ORDER BY WTIDate
```

	WTIDate	WTIValue
1	1986-01-02	25.5600
2	1986-01-03	26.0000
3	1986-01-06	26.5300
4	1986-01-07	25.8500
5	1986-01-08	25.8700
6	1986-01-09	26.0300
7	1986-01-10	25.6500
8	1986-01-13	25.0800
9	1986-01-14	24.9700
10	1986-01-15	25.1800
11	1986-01-16	23.9800

As we can see, extracting API data in PowerShell does not require too much, as this will look similar even if we were to obtain other JSON documents. We will find that most of the changes come when we choose if we want (1) all the fields in a document, (2) some of the fields, (3) fields in one or more subdocuments, (4) and a few other possible combinations. In those cases, it's matter of understanding the JSON (or XML/HTML) structure and getting what's needed - the way that we access and add the data remains the same. Finally, remember that developers can use data tables if they prefer and convert accordingly; different environments have different limits, so building something that matches our environment will reduce the probability of performance issues.

Next Steps

- Study and review an API that you'd like to extract data from; keep in mind that some APIs may charge, so for testing, pick a free one (when it's operating, Bitstamp is a good starter).
- Test converting a JSON string and obtaining the fields you want.

Last Updated: 2015-02-03

About the author

(/sqlserverauthor/92/tim-smith/) Tim Smith works as a DBA and developer and also teaches Automating ETL on Udemy.

View all my tips (/sqlserverauthor/92/tim-smith/)

Related Resources

- More SQL Server DBA Tips... (/sql-server-dba-resources/)


feeds.feedburner.com/MSSQLTips-LatestSqlServerTips) (/)

MENU

Resources

- Tutorials (/sql-server-tutorials/)
- Webcasts (/sql-server-webcasts/)
- Whitepapers (/sql-server-whitepapers/)
- Tools (/sql-server-tools/)

Search

- Tip Categories (/sql-server-categories/)
- Search By TipID (/search-tip-id/)
- Authors (/sql-server-mssqltips-authors/)

Community

- First Timer? (/learn-more-about-mssqltips/)
- Pictures (/mssqltips-community/1/)
- Contribute (/contribute/)
- Event Calendar (/sql-server-event-list/)
- User Groups (/sql-server-user-groups/)
- Author of the Year (/mssqltips-author-of-year/)

More Info

- Join (/get-free-sql-server-tips/?ref=JoinFooterMenu)
- About (/about/)
- Copyright (/copyright/)
- Privacy (/privacy/)
- Disclaimer (/disclaimer/)
- Feedback (/feedback/)
- Advertise (/advertise/)