Getting JSON Lines from an API Call into a CSV File via UNIX Command Line

Posted on October 3, 2022

WhoisXML API offers, via API calls, a lot of information on domains or IP addresses that is very useful for a number of applications, including cybersecurity investigations, domain appraisal or system administration. Imagine that one is given a long list of domain names, and one is interested in each domain's details, e.g., WHOIS or website categories, etc. In each API call, a single domain name is sent, and its data is returned in JSON format. (Some of the APIs of WhoisXML API offer bulk lookups, too, but we will not deal with this possibility here.) To get an overview of the data, analysts often prefer to have everything in a single csv file; one line for each domain. This can be opened by popular office spreadsheets like Excel or LibreOffice Calc, or sent to other kinds of processing.

In this blog post, we show a way to extend a list of domain names with additional data from individual API calls from WhoisXML API services. We will work on a Linux machine in BASH, hence, the same can be done on MacOS X, or on a Windows using Windows Subsystem for Linux. A basic level of UNIX command-line expertise is assumed. We will use the Website Categorization API with a list of domains, but our recipe will be applicable for all the other APIs and also with a list of URLs or IP addresses.

1. The task

Suppose we are given a list of domains (or URLs), in a text file, domains.txt, which looks like this:

    whoisxmlapi.com
    bbc.co.uk

---

amazon.com

google.com

Of course, the list can be much longer, this is just an example. For each domain, we want to see the metadata of the websites hosted on them, as well as the tags, as well as the top Internet Advertising Bureau (IAB) categories applicable to them. The Website Categorization API will provide us with all this information; it will give us results starting like this:

```
{
 "categories": [
 {
 "tier1": {
 "confidence": 0.9499015947838411,
 "id": "IAB-596",
 "name": "Technology & Computing"
 },
 "tier2": {
 "confidence": 0.8420597541031617,
 "id": "IAB-618",
 "name": "Information and Network Security"
 }
 },
 {
 "tier1": {
 "confidence": 0.9499015947838411,
 "id": "IAB-596",
 "name": "Technology & Computing"
 },
 "tier2": {
 "confidence": 0.6916495127489835,
 "id": "IAB-623",
 "name": "Email"
```

```
    }
  },
```

and ending like this:

```
  {
  "tier1": {
  "confidence": 0.9499015947838411,
  "id": "IAB-596",
  "name": "Technology & Computing"
  },
  "tier2": {
  "confidence": 0.5670964415978328,
  "id": "IAB-600",
  "name": "Computer Networking"
  }
  }
  ],
  "domainName": "whoisxmlapi.com",
  "websiteResponded": true
```

(We have omitted a number of categories from the middle to save some space here.)

Recall that we want to have this in a csv file as a single line. And here is the problem we are facing: the API returns a hierarchy of fields, and there is a variable-length list involved, too. While in JSON, lists and hierarchy are natural, we just want everything in a single series of records, as structured as possible, though. We need a way to "flatten" our JSON. Before resolving this issue, however, let us get the data for all the domains.

2. Getting data

Throughout this demonstration, we will be working in a terminal in BASH. To call the API, let's

have the API key. If you don't have one, it is obtained after registration. (A free subscription is also available if you want to give it a try.) We will keep the API key in an environment variable:

```
export APIKEY=xxx
```

(replace xxx with your actual API key).

First of all, let's try and see if the API actually works with our settings, let's check whoisxmlapi.com:

```
curl --get "https://website-categorization.whoisxmlapi.com/api/v2?apiKey=$APIKEY&domainName=wh
```

We will be using curl for doing the API calls; the data for whoisxmlapi.com should be seen in the terminal.

The simplest way to get the data for all the domains is just to loop through them and use curl to save the result into a file for further processing:

```
cat domains.txt | while read d
do curl --get "https://website-categorization.whoisxmlapi.com/api/v2?apiKey=$APIKEY&domainName=
   |tee categorization_$d.json
done
```

We use tee to make the result visible on the screen while running. This will give us a file, e.g. categorization_whoisxmlapi.com.json, for each of the domains, with the JSON content returned by the API. The contents of these are to be transformed into lines of a single csv file.

Before turning our attention to the conversion, however, let us note that the lookup of these domains could have been done using multiple jobs in parallel. Having GNU parallel installed, the same files can be generated like this:

```
cat domains.txt  | \
```

```
parallel --jobs 4 \
'curl --get "https://website-categorization.whoisxmlapi.com/api/v2?apiKey=$APIKEY&domainName="{}
```

We have set the number of jobs to 4, so the speed will be about 4 times faster. You may experiment with bigger values, but keep in mind that the APIs have a throttling limit, which is 30 requests per second by default (no limit in enterprise subscriptions).

Having the data in JSON format at hand, let us convert them now to csv.

3. Flatten JSON lines to CSV

To convert JSON lines to fields with records, jq is frequently used. It requires, however, a nontrivial specification of the conversion. Here we adopt a more general approach: we want to flatten the hierarchical structure of our JSONs as simply as possible, i.e., without making too many considerations about the fields. (We shall see that, unfortunately, we cannot completely avoid this, though.) As expected, we are not the first to have this requirement; a possible solution is json2csv. Having Node v12 or higher on our machine, this can be installed by doing

```
npm install -g json2csv
```

We can try now converting one of our JSON files to csv:

```
json2csv --flatten-objects --flatten-arrays -i categorization_whoisxmlapi.com.json
```

which will write a nice csv with a header line to the standard output. It will turn JSON objects into separate fields and flatten the lists also to multiple fields; we will have fields like this:

```
"categories.0.tier1.confidence","categories.0.tier1.id","categories.0.tier1.name","categories.0.tier2.con
"categories.0.tier2.id","categories.0.tier2.name",
```

```
"categories.1.tier1.confidence","categories.1.tier1.id","categories.1.tier1.name","categories.1.tier2.con
"categories.1.tier2.id","categories.1.tier2.name"
```

If we now create a single file with all the JSONs, one each line (i.e. a jsonl file):

```
cat categorization_*.json > categories_all.jsonl
```

We expect that using this as json2csv-s input, it will create the desired csv file. Well, almost… At the time of writing, json2csv is missing a feature: when converting JSON arrays, it is not able to properly handle their variable lengths when converting multiple csv lines. In our case, for instance, amazon belongs to 9 categories, whereas WhoisXML API to only 7 of them. When converting a file with json2csv, it will use the first line of the jsonl input to determine the field structure, so if eventually, whoisapi's line precedes amazon's on the input, the two last list elements will disappear.

In fact, flattening variable length arrays to csv without losing information is not a trivial matter. What one would like to see is a number of csv fields that correspond to the maximum length of the given array in the data set, left blank for records that do not fill it. But how do we know the maximum length? It is not possible to determine it in advance, before reading all records. So we could either store the whole data set in the memory or read all the records sequentially twice: first to find out the maximum length of array(s), second to perform the conversion. (Here, we have a single array, but in some of the APIs, like the Website Contacts API there can be multiple independent arrays.)

Here we adopt a simpler approach: we do a rough estimate of maximum 10 array elements. Of course we will lose those beyond it, but if amazon has 9, this should be sufficient. Unfortunately, when using json2csv the fields for flattened arrays can only be controlled by the explicit listing of fields. So we write a small script that produces the respective list:

```
#!/bin/bash
NMAX=9
printf '%s' "domainName,websiteResponded"
```
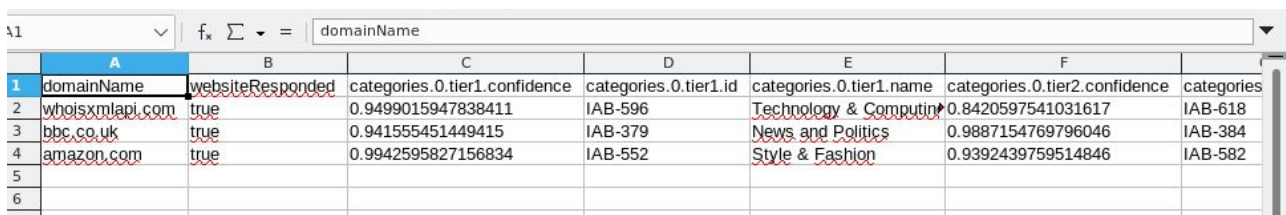
```
for i in $(seq 0 $NMAX);do
   printf \
',categories.%d.tier1.confidence,categories.%d.tier1.id,categories.%d.tier1.name,categories.%d.tier2.
   $i $i $i $i $i $i
done
```

Note that to have 10 fields we have set NMXAX=9 as the fields are numbered from 0. Also, we have put the domain name and the boolean value of whether the website has responded in the first place. By default, it would appear at the end of the row as in the JSON entry; a benefit of having to list the fields is that we can also modify their order.

Now finally, we can do our conversion:

```
json2csv --flatten-objects --flatten-arrays \
   -f $(./echo_categorization_header.sh) \
   -i categorization_all.jsonl \
   -o categorization_all.csv
```

This results in the file categorization_all.csv: a neat csv file with a proper header, which is ready for further analysis, e.g. using office software such as Excel or LibreOffice:

4. Summary

We have presented, through an example, a way of producing a csv file from individual API calls with a parameter coming from a file, and resulting in JSON output of the same structure. There are many ways of doing it, e.g., via pandas using Python, or ODBC-based approaches in excel. Here we have shown a very simple approach and use open source tools in a bash command line. In this demonstration, we were using WhoisXML API's Website Categorization API, but it can be

considered as a general recipe in many similar situations.