Download WhoisXML API daily data ASAP part 1: RSS-activated download

Posted on June 11, 2021

This technical blog aims to demonstrate how to download data from WhoisXML API's daily data feeds right after their publication. Obtaining lists of newly registered domains and their WHOIS data can be critical in many applications. (We will showcase such an application in the second part of this blog.) Recently an RSS feed has been introduced for the service that informs about data publication immediately, which makes this easy. As a demonstration, we shall go through a particular task: download the list of domains newly registered in the .com top-level domain (TLD).

We will use a Linux system and its understanding requires intermediate programming and command-line skills. We will use BASH but it is also easy to modify for zsh, which is the default on Mac OS X. We assume Python, with pip and virtualenv.

Principle

WhoisXML API provides accurate lists of domains registered, dropped, and modified second-level domains in generic top-level domains and their WHOIS data once a day. These data feed services are documented in detail in a comprehensive manual. As said before, now we opt for the list of newly registered domains in .com. More precisely, we shall get a list of domains that were not present in the domain name system the day before; they have just appeared on the given day.

The data we need are published in a data feed named domain_names_new once a day, typically no later than 21:00 GMT/UTC. But why "no later"? This is there because the actual time of availability of this file depends on many random factors. In many applications, however, time does matter: a domainer may, e.g., miss a great deal because of the delay, but also because of the inaccuracy of data. So how do you download data ASAP then?

The trick is, there is an RSS feed in each data feed's directory. Namely, in the subdirectory status, which holds many useful files like the list of TLDs supported by the given data feed, we have the file download_ready_rss.xml. This gives immediate information on the completion of a data download.

A title of an entry looks like this:

domain_names_new csv 2021_03_01 2021-03-01 19:51:51 UTC

i.e., it has the data feed name, the data format, the date for which the data are prepared, and the UTC time of the data creation, which is actually the very time when the RSS entry appears. By this time, all the data for all TLDs in the given feed in the given format are complete and ready for download. The body of the message holds all this information in JSON format to facilitate machine parsing; in our example:

```
{"data_feed": "domain_names_new",
"format": "csv",
"day": "2021-03-01",
"available_from":
"2021-03-01 19:51:51 UTC"}
```

So, the algorithm to get data ASAP is fairly simple:

- Watch the RSS feed regularly
- If a new entry appears, download the data
- go to 1.

Data access

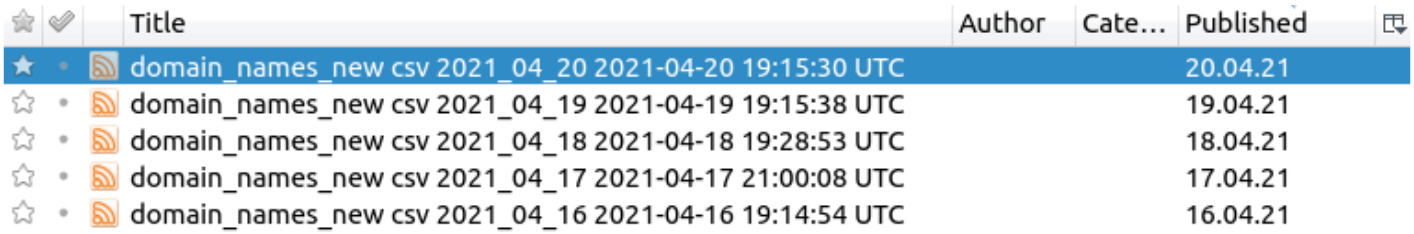
To follow our experiments you need a subscription to WhoisXML API's Newly Registered Domains data feed. As we are dealing with names of new domains only, the "lite" subscription plan is

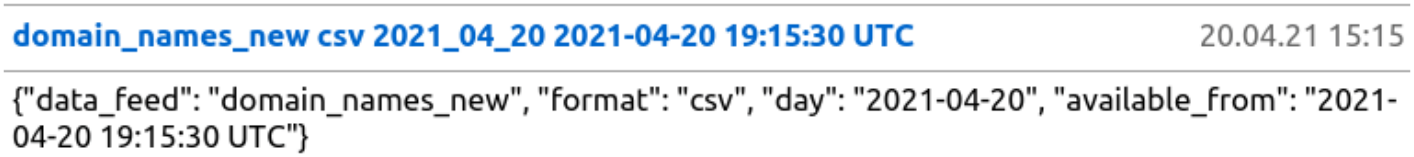
sufficient, but the more extensive plans also include the required data set. Upon subscription, you will receive a "key", which will be the username and password for plain http authentication. (If you have a legacy subscription to the "domain_names_new" data daily data feed, with a separate username and password, you can also follow the implementation, we will comment on the differences later.)

RSS feed

The notification feed is located at the following URL:
https://newly-registered-domains.whoisxmlapi.com/datafeeds/Newly_Registered_Domains/lite/domain_names_new/status/downlo

Replace "lite" with the name of your subscription plan here. Legacy users will find this under https://bestwhois.org/domain_name_data/domain_names_new/status/download_ready_rss.xml accessible with their username and password. When viewed with an interactive RSS reader, the feed looks like this:





We will also need rsstail, a command line utility that can monitor an rss feed. It is available from here:
https://github.com/flok99/rsstail

On Debian-flavor systems, it can be easily installed from a standard package:

```
sudo apt-get install rsstail
```

Data sets

To download the data we will use the Python downloader script provided by WhoisXML API. It is time to start our particular implementation, so let's open a shell prompt and do it:

```
mkdir daily_words
cd daily_words
git clone https://github.com/whois-api-llc/whois_database_download_support
mv whois_database_download_support/whoisxmlapi_download_whois_data/ \ downl
```

Our project will reside in the "daily_words" subdirectory. (The reason for this choice of a name will be clear in the second part of this blog; you may opt for any other name.) So far, we have git-cloned WhoisXML API's client-side script support package, and we have picked the script we need and got rid of the rest.

It is time now for creating a Python virtual environment to avoid interference with any other Python application. (You may skip this step but it does have its benefits.) We install the required packages, too:

```
python3 -m venv daily_words_venv
source ./daily_words_venv/bin/activate
pip3 install wheel
pip3 install -r download_whois_data/requirements.txt
```

Now we are in the position to try downloading a data set. An example command-line is:

```
mkdir data
./download_whois_data/download_whois_data.py \
   --feed domain_names_new \
   --tlds com  \
   --dataformats csv \
   --startdate 20210420 \
   --plan lite \
   --password $MY_KEY \
   --output-dir ./data
```

We have created a subdirectory named "data" and have downloaded the list of new domains in com for 2021-04-20 into it. (Replace "lite" with your actual subscription plan, "$MY_KEY" with your access key, and the date, e.g. 5 days ago.) The actual data set is in the subdirectory

```
data/domain_names_new/com/2021-04-20/add.com.csv
```

in our case.

Completing the data download implementation

Having set up and tested all the requirements, we are ready to implement the script that will download the data as soon as they're available. Our script, named get_com_daily_new.sh reads

```
#!/bin/bash

MY_PLAN=lite
MY_KEY=MY_KEY

PROJECT_DIR="$( cd "$(dirname "$0")" >/dev/null 2>&1 ; pwd -P )"
cd $PROJECT_DIR

source ./daily_words_venv/bin/activate

(rsstail -n1 -u \
"https://${MY_KEY}:${MY_KEY}@newly-registered-domains.whoisxmlapi.com/dataf
> rss_output.txt & echo $! >&3) 3>rsstail.pid

sleep 5

last_date=$(cat last_date.txt)
echo "---------------"
date
echo "Last download for day: $last_date"
```

```
available=$(tail --lines 1 < rss_output.txt | cut -d " " -f 4)
echo "Last available data for day:" $available
if [[ $last_date == $available ]];then
   echo "We already have data for $available"
else
   echo "Downloading for $available"
   datearg=$(echo $available | sed -e s/_//g)
   cd ./download_whois_data
   ./download_whois_data.py --feed domain_names_new \
      --tlds com \
      --dataformats csv \
      --startdate $datearg \
      --plan $MY_PLAN --password $MY_KEY \
      --output-dir ../data
   cd ..
   echo $available > last_date.txt
fi

kill $(cat rsstail.pid)
```

Note: please configure the variables MY_PLAN and MY_KEY to make it work. As a brief explanation: we change into the script's directory because we plan to run this script from crontab repeatedly, and then we activate our virtual environment. To see new entries in the RSS feed, we use rsstail, which is designed for interactive use; it does not exit after displaying the last posts in the feed. So, we start it in the background, and capture its output into rss_output.txt (then we wait for 5 seconds, which should be enough to get it), and write its PID to rsstail.pid; we kill it at the end of the script.

Then we compare the date of the newest daily data set available with the one we already have. This latter date is kept in last_date.txt, so removing this file will forcibly repeat the download process.

Running the script and looking at the data directory, we can observe the newly downloaded data set available for further processing. What remains is to run this script regularly to get the updates ASAP. In our demonstration, we opt for a frequency of 1 hour, so if the data are published, we will know about it in less than an hour. So, we add the following line to our user's crontab (note that itis a single line):

```
0 * * * * /FULLPATH/daily_words/get_com_daily_new.sh >>  /FULLPATH/daily_wo
```

Here FULLPATH is to be replaced with the path to the processing script. We direct stdout and stderr to a log file to verify the activity of the script. In critical applications you may want to opt for a shorter periodicity of running the script, but it is not reasonable to go below about 20 minutes.

Now we have the fresh data ASAP, and this solution can be used for all those gTLD daily data feeds that provide RSS notifications. We are aware that this is a very important use case for many of our customers. In the continuation of this 2-part post we shall describe one such simple application.