# Using APIs to Access Data on Stocks, Social Media, News and More

Robert P. Schumaker*

Professor - Computer Science Dept., University of Texas at Tyler

Director - Data Analytics Lab, University of Texas at Tyler

Consultant - ORS Research Design and Data Analysis Lab

February 16, 2022

## Contents



*rob.schumaker@gmail.com

# 1 Process Overview

This document will provide a demonstration of several data collection programs that researchers can use for their own projects. In particular, readers will learn how to collect per minute stock prices, news articles tied to specific stock tickers, and social media posts from Twitter.
Collecting web data has come a long way in the last 30 years. From copy/pasting, to writing web scrapers and parsers, to using specific interfaces (APIs) to collect precise data.
This tutorial will explore the history and use of APIs and provide a brief tutorial for certain data collections.

# 2 API Overview

Collecting real-time data can be an exciting and richly rewarding academic endeavor. There are plenty of free data feeds available and coupled with modern data extraction techniques, can provide research opportunities that were previously unobtainable. In this overview we will barely scratch the surface of obtaining web data.
API, or Application Program Interface, is a way of connecting computers and programs. The term was originally coined in the 1940s and used to describe early software programming libraries. The definition has evolved and now describes a networked computer method of easily sharing data. The API involves a data provider who governs the individuals and/or entities that have data access, how the data is accessed as well as how much data can be accessed. Because of their proprietary nature, each data provider will have their own unique way of requesting and receiving data. While it may be unique between data providers, the process is roughly the same.

- Authentication (optional)
- Request data
- Receive data
- Parse data

Let's explore each area in more detail.

## 2.1 Authentication

Authentication is the process of checking a user's access credentials. This step can be optional by some data providers or be used to allow different levels of data access (guest vs authenticated user). Methods of authentication could include username/password combinations or the use of generated tokens (e.g.; OAuth), or an api-related key. This step verifies the user is credentialed and known.
Some data providers will also include Authorization along with authentication. Authorization establishes the level of data access or grants access to specific data. These two concepts are different as a user could be authenticated (yup you logged in) but not authorized (nope, you have no access here).
The data provider will let you know whether authentication is necessary.

## 2.2 Data Request

The method of requesting data from the data provider will vary between providers. Most use some type of keyword phrasing with a proper syntax. Data requests can be done through url requests or specific keywords directed towards a data provider's listener. For example, requesting flight prices from Skyscanner can be done via a url request formatted like this:
*https://partners.api.skyscanner.net/apiservices/browsequotes/v1.0/{country}/{currency}/{locale}/{originPlace}/{destinationPlace}/{outboundPartialDate}/{inboundPartialDate}?{apiKey={apiKey}}*, where {apiKey} is your authentication token.
Likewise, requesting weather forecast data from OpenWeather is also done via a url request formatted like this:
*https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={apiKey}.*
Contrast this to Twitter's firehose approach where data requests are directed to a listener and requests must be made programmatically.
Url-based data requests do not necessarily require software programming experience, although it could be helpful. A carefully crafted Excel spreadsheet could perform url-based data collection using smart-tags or minimal VBA programming. Whereas listener-based data collection will require a program to perform the authentication, data negotiation and data handling operations. For well-known APIs there is generally software available either from the data provider or third-parties.

The exact syntax and use of the data request will be defined by the data provider in a data dictionary or API documentation.

### 2.3 Data Receipt

Once a proper data request is made, the data provider will return data to the requester. In the case of a url-based request the transaction is considered closed and any further requests will require the authentication mechanism again if applicable. In the case of a listener-based request the transactional data flow continues until either side severs the connection.
The data that is returned will generally fall into one of three categories: unformatted text, xml or json. Although other boutique data schemas exist, these three are the most typical and easiest to handle. Unformatted text is typical for simple requests where only one response is given. XML, or eXtensible Markup Language, is an older standard that uses angled brackets with a data hierarchy. The following is an XML example:

```
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>Two of our famous Belgian Waffles with plenty of
      real maple syrup</description>
    <calories>650</calories>
  </food>
  <food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>Light Belgian waffles covered with strawberries
      and whipped cream</description>
    <calories>900</calories>
  </food>
</breakfast_menu>
```

The most common form of data exchange is using JSON. JSON, or JavaScript Object Notation, originally was created for the JavaScript programming language in 1999, but eventually separated and became its own language-independent standard in 2013. Its structure is similar to XML but with less overhead. The following is a JSON example:

```
{
  "breakfast_menu": {
    "food": [{
      "name":        "Belgian Waffles",
      "price":       "$5.95",
      "description": "Two of our famous Belgian Waffles with
                      plenty of real maple syrup",
      "calories":    "650"
    },
    {
      "name":        "Strawberry Belgian Waffles",
      "price":       "$7.95",
      "description": "Light Belgian waffles covered with
                      strawberries and whipped cream",
      "calories":    "900"
    }]
  }
}
```

Regardless of the structure the data provider uses to return the data, there are techniques to extract what you need.

### 2.4 Data Parsing

Turning returned data into something useable to the researcher is its own art form. With unformatted text, the entirety of the response is typically useable. With XML and JSON a bit of post-processing is necessary. Luckily there are many software libraries, stand-alone programs, or even online extraction tools that can do these tasks for us. It isn't perfect and may require some further tweaking depending on your needs.

One such example for extracting XML data is *https://onlinexmltools.com/convert-xml-to-text*. This tool strips the xml tags and leaves the entirety of text. Not great if you need specific data. The typical method is to create XSLT stylesheets and extract the data using that. However, Microsoft Excel offers some XML conversion functionality. For more information on converting XML with Excel please see *https://stackoverflow.com/questions/9925108/extract-data-fields-from-xml-into-excel*.

An example for extracting JSON data is *https://jsonpathfinder.com/*. This online resource can both interrogate JSON output but also search and return specific data using the "Path" function. For more information on using

JSON Path, please see *https://www.toolsqa.com/rest-assured/jsonpath-and-query-json-using-jsonpath/*.
Worst case scenario the data is in a non-standard form in which a programmer must create a customized parser to extract the necessary data. Luckily we don't run into that too often nowadays.

## 3 Some Popular APIs for Data Collection

A quick Google search can turn up hundreds of available APIs. Some better than others. Here is a quick listing of a variety of APIs, what they offer, and a link to their documentation. Hopefully one of them inspires you.

- API-Football - soccer data on 905 leagues and cups
  https://www.api-football.com/documentation-v3
- API-NBA - basketball data
  https://rapidapi.com/api-sports/api/api-nba
- Brave NewCoin - cryptocurrency data
  https://bravenewcoin.com/developers
- ExchangeRates - currency exchange data
  https://exchangeratesapi.io/documentation/
- Facebook - social media data
  https://developers.facebook.com/docs/
- FBI Crime Data - find crooks
  https://api.data.gov/docs/fbi/
- Financial Times - financial news data
  https://developer.ft.com/portal/docs-api-reference
- IMDB - movie data
  https://developer.imdb.com/documentation/
- NewsAPI - worldwide news stories
  https://newsapi.org/s/google-news-api
- OpenWeather - weather data
  https://openweathermap.org/api

- Parler - social media data
  https://github.com/KonradIT/parler-py-api
- Pinterest - social media data
  https://developers.pinterest.com/
- Skyscanner - flight data
  https://skyscanner.github.io/slate/#api-documentation
- SportRadar - cricket sports data
  https://developer.sportradar.com/docs/read/cricket/Cricket_v2
- SportsdataIO - baseball data
  https://sportsdata.io/developers/api-documentation/mlb
- VIN Decoder - vehicle manufacturing data
  https://vpic.nhtsa.dot.gov/api/
- Yahoo Finance - stock price data
  https://www.yahoofinanceapi.com/

# 4 Tutorial - Collecting Real-time Data through APIs

The following tutorials assume that you have a database environment to store your data and a Java Runtime Environment.
If you do not have a database environment, please refer to the prior tutorial *Setting up an Amazon Web Services (AWS) and RStudio Data Connective Environment* and perform sections 3 to 3.3.2. Please see the ORS Data Science Resources webpage at *https://uttyler.edu/research/ors-research-design-data-analysis-lab/resources/data-science* or reach out to the author for the tutorial.
If you do not have a Java Runtime Environment, please refer to the prior tutorial *Collecting Real-time Tweets through Twitter's Firehose* and perform section 3.4.1. Please see the ORS Data Science Resources webpage at *https://uttyler.edu/research/ors-research-design-data-analysis-lab/resources/data-science* or reach out to the author for the tutorial.

## 4.1 Stock Quote and Stock News Setup

These instructions will get your research environment ready to collect per-minute stock quotes and/or stock news. Once setup it is recommended to

collect data no more often than once per day.

### 4.1.1 Database Configuration using mySQL Workbench

The following instructions will setup your database for both Stock Quotes and Stock News. If you will be collecting both, you only need to perform these instructions once.

1. Open mySQL Workbench
2. Click the plus sign in a circle to create a connection to the database server
3. For **Connection Name** pick a name for your connection. The name doesn't really matter, it's to help you find it
4. For **Hostname** on a local machine, enter 127.0.0.1
5. For **Hostname** on an Amazon RDS instance, enter the RDS endpoint (e.g., oceanplatform0.cdb7tnix15tn.us-west-2.rds.amazonaws.com)
6. For **Username**, enter *root*
7. Click **Test Connection** and enter the mySQL admin/root password
8. If you successfully made the mySQL connection, click Ok to exit the **Setup New Connection** dialog box
9. If you were unsuccessful making the mySQL connection, find a file named *mysqld.cnf*, open it for editing, put a # symbol in front of *bind-address = 127.0.0.1*, and either restart the mysql process or reboot the computer
10. Click your new connection under **mySQL Connections**
11. Click **File**, **Open SQL Script** and open *SP500_ dbsetup.sql*
12. Modify lines 43-547 to reflect the stock ticker symbols you wish to collect quotes. Each line represents one company. Feel free to expand/reduce the number of companies. The only required field is the first one (CompSymbol) which is the stock ticker. The other fields can be left blank, just keep the commas intact.

13. Modify near line 7346, select a username/password for SP500
14. If you change the username, be sure to rename 'webwrite' to your new username in the five lines near line 7347
15. Click the lightning bolt to execute the script. If everything goes well you should have all green checkmark circles in the **Action Output** window

### 4.1.2 Scripting Configuration

The Stock Quote and Stock News application is text-based and requires no installer. However, there will be some configuration necessary.
The configuration can initially seem overwhelming. However, a careful review of these instructions should be helpful. If you find yourself stuck, reach out to the author.
The configuration file services both Stock Quotes and Stock News, and is designed to run sequentially. It has several major parts:

1. File path to the SP500.Quotes folder
2. File path to the SP500.News folder
3. Location of your java interpreter (from OpenJDK you installed earlier)
4. Classpath to the program libraries. The path should be identical for all of the jar files. Just a lot of copy/pasting or Edit/Replace
5. The package.program to run, in this case it is sp500.quotes.QuotesMain and sp500.news.NewsMain respectively
6. Database location and schema
7. Database username and password

What to do...

- Download the *SP500.zip* file from the ORS Resource webpage
- Unzip the *SP500.zip* file and move the SP500 folder to a directory location of your choosing
- Open the *SP500* folder and open the *SP500.sh* for editing. It is a text file, so just about any text editor will work

- The first line is **#!/bin/bash**. This has special meaning in linux-based operating systems but not so much in Windows. If your computer is Windows-based, remove this line

- The next two lines are for *Stock Quotes*. If you are not collecting quote data, remove these two lines

- The last two lines are for *Stock News*. If you are not collecting news data, remove these two lines

- In the unmodified version of the script, lines 2 and 4 set the file path to the SP500.Quotes and SP500.News folders respectively. My file path to SP500.Quotes is */home/rschumaker/GoogleDrive/OceanPlatform/SP500.Quotes*. On my Windows machine it is *C:\Users\robsc\Desktop\SP500\SP500.Quotes*. You need to find the exact path of your SP500.Quotes folder. Modify the file path for both SP500.Quotes and SP500.News (if applicable)

- In the unmodified version of the script, lines 3 and 5 run the Quotes and News scripts.The first section */usr/lib/jvm/jdk-17.0.2/bin/java* is the file location of your java interpreter. This file path **must be changed**. You must find the java interpreter and copy it's path. On my Windows machine I would change it to *"C:\Program Files\jdk-17.0.2\bin\java"* with the double-quotes

- The next section is the path to the SP500.Quote and SP500.News folders respectively. My classpath for SP500.Quotes in linux is */home/rschumaker/GoogleDrive/OceanPlatform/SP500.Quotes/*. On my Windows machine it is *C:\Users\robsc\Desktop\SP500\SP500.Quotes*. You need to find the exact path to your respective folders. Replace **all** instances of /home/rschumaker/GoogleDrive/OceanPlatform/ with your path

- If you are using a Windows machine, change all colons in the classpath area to semicolons. Please do not use **Replace All** as there are colons outside of the classpath area

- Next is the location of your database and schema. Enter the IP address or endpoint information to your database. If you are using a schema different from the *SP500_ dbsetup.sql* schema, please enter it after the *3306:/*

- Enter your database username and password. You set these values in *SP500_ dbsetup.sql* near the bottom of the script
- Save the file and close it
- If you are using a Windows machine, rename the file extension to **.bat**

#### 4.1.3 Collecting Stock Quotes and Stock News

Congratulations, you made it to this point. Assuming that the database server is correctly setup, OpenJDK is installed, the file path configuration is perfect and you have said your required 1,000 Hail Mary's, SP500.sh is ready to run.

1. On linux-based machines, use terminal to navigate to the SP500 directory and type *sh SP500.sh*
2. On Windows-based machines, use the command prompt to navigate to the SP500 directory and type *SP500.bat*
3. You should now see a line **Acquiring Quotes for...** and your stock ticker. To avoid blacklisting by the stock ticker service, there is a one minute pause between tickers.
4. The most common error that could occur is the **Are you a robot**. This means that you have stopped/started the program too quickly, someone else at the University is using the program, or Bloomberg's anti-robot algorithms got around my anti-anti-robot evasion routines in my program. It happens. What tends to work best is to run the program at home rather than at the University.
5. Following stock quote data gathering, the script will then collect the 20 most recent news articles for each stock ticker. Again there will be a pause between each article, but not as great a pause as before.

### 4.2 Twitter

To learn how to gather Tweet data, please refer to the prior tutorial *Collecting Real-time Tweets through Twitter's Firehose*. Please see the ORS Data Science Resources webpage at *https://uttyler.edu/research/ors-research-design-data-analysis-lab/resources/data-science* or reach out to the author for the tutorial.