

Stata-Python API for bulk data download: Example with UN Comtrade

Ka Lok Wong (Steve)
Geneva Graduate Institute, Switzerland



November 18, 2022

- Increasing availability of trade data through public domains
 - 40+ years, 6-digit (~5,300 products)
 - So as bulk data size
- Papers made use of Comtrade data/ [BACI dataset on CEPII](#):
 - [Gaulier and Zignago, 2012](#);
 - [Hausman and Hidalgo, 2011](#);
 - [Stadler et al, 2018](#)
- Downloading and importing 40 Gb+ CSV files is tedious and **very prone** to human error

What is not an API?



UN Comtrade Database

Extract data ▾

Data Availability ▾

Knowledge base

API portal

1. Type of product & Frequency

Type of product

 Goods Services

Frequency

 Annual Monthly

2. Classification

HS

 As reported 92 96 02 07 12 17 22

SITC

 As reported Rev. 1 Rev. 2 Rev. 3 Rev. 4

BEC

 BEC

3. Select desired data

Periods (year)

All or a valid period. Up to 5 may be selected.

Reporters

All or a valid reporter. Up to 5 may be selected. All may only be used if a partner is selected.

Partners

World, All, or a valid reporter. Up to 5 may be selected. All may only be used if a reporter is selected.

Trade flows

All or select multiple trade flows.

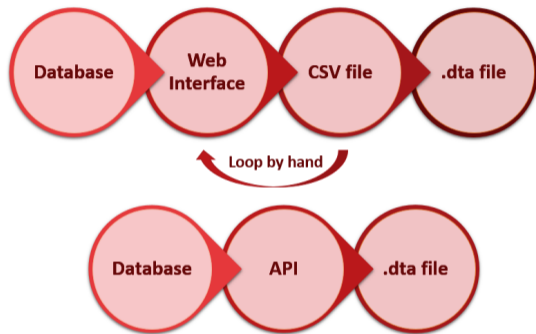
HS (as reported) commodity codes

All, Total, AG[X] or a valid code. Up to 20 may be selected. If you know the code number, e.g. 01 - Live animals, type 01. To search by description type a word, e.g. rice.

4. See the results

[Get data »](#)[Download CSV !\[\]\(a3a7cedfdf1046b3481312a851ed6b42_img.jpg\)](#)[More information about data !\[\]\(175406b766831414c887aca642d4d7a3_img.jpg\)](#)

Issues opening CSV in Excel? See this Microsoft how-to.



- **Alternative:** automatizing the tasks from the data sourcing stage, i.e. **API**
- API stands for **Application Programming Interface**
- It allows two applications to communicate with each other using requests and responses
- API documentation is therefore a **map** to find the way through the maze

Benefits: {
Time saving
Memory saving
Automation to avoid human error

What is an API? (cont'l)

| UN Comtrade Database | | Extract data ▾ | Data Availability ▾ | Knowledge base | API portal | | | | |
|----------------------|--------|---------------------|---------------------|----------------|-------------------|---|-------------|---|---|
| 2020 | Export | Rwanda | China | <u>TOTAL</u> | \$8,356,692 | 0 | No Quantity | 0 | 0 |
| 2020 | Export | Saudi Arabia | China | <u>TOTAL</u> | \$8,181,515,875 | 0 | No Quantity | 0 | 0 |
| 2020 | Export | Tajikistan | China | <u>TOTAL</u> | \$34,411,308 | 0 | No Quantity | 0 | 4 |
| 2020 | Export | Trinidad and Tobago | China | <u>TOTAL</u> | \$154,117,034 | 0 | No Quantity | 0 | 0 |
| 2020 | Export | Egypt | China | <u>TOTAL</u> | \$603,114,245 | 0 | No Quantity | 0 | 4 |
| 2020 | Export | Albania | China | <u>TOTAL</u> | \$46,049,692 | 0 | No Quantity | 0 | 0 |
| 2020 | Export | Angola | China | <u>TOTAL</u> | \$13,134,198,358 | 0 | No Quantity | 0 | 4 |
| 2020 | Export | Azerbaijan | China | <u>TOTAL</u> | \$432,760,538 | 0 | No Quantity | 0 | 0 |
| 2020 | Export | Argentina | China | <u>TOTAL</u> | \$5,244,394,119 | 0 | No Quantity | 0 | 0 |
| 2020 | Export | Australia | China | <u>TOTAL</u> | \$100,085,567,846 | 0 | No Quantity | 0 | 4 |
| 2020 | Export | Austria | China | <u>TOTAL</u> | \$4,467,013,853 | 0 | No Quantity | 0 | 4 |
| 2020 | Export | Bahamas | China | <u>TOTAL</u> | \$56,013 | 0 | No Quantity | 0 | 4 |
| 2020 | Export | Bahrain | China | <u>TOTAL</u> | \$131,825,462 | 0 | No Quantity | 0 | 0 |
| 2020 | Export | Armenia | China | <u>TOTAL</u> | \$265,212,059 | 0 | No Quantity | 0 | 0 |
| 2020 | Export | Barbados | China | <u>TOTAL</u> | \$2,666,403 | 0 | No Quantity | 0 | 4 |
| 2020 | Export | Belgium | China | <u>TOTAL</u> | \$7,628,839,203 | 0 | No Quantity | 0 | 4 |

Showing 1 to 25 of 145 entries

First Previous **1** 2 3 4 5 Next Last

Estimated quantity/netweight shown in italics.

Flag refers to quantity/netweight estimation:

0 = no estimation, 2 = quantity, 4 = netweight, 6 = both quantity and netweight

Modify selection ▲

Download CSV 📄

[View API call](#) | [API documentation](#)

```
/api/get?max=502&type=C&freq=A&px=HS&ps=2020&r=all&p=156&rg=2&cc=TOTAL
```

- 1 Identify the data of interest
- 2 Understand the API call
- 3 Check the API Link
- 4 Test run with 1 year
- 5 Loop it through years
- 6 Some final notes

Trade data as an example



Step 1: Identify the data of interest

- Country-pair: country i to China, for all countries in the world
 - Direction: Export
 - Frequency: Annual
 - Time period: 2000 to 2021
 - Classification: All 2-digit SITC Rev. 2
- 1 Input the according info on [Comtrade - Get data](#)
 - 2 “**View API call**” in the bottom
 - 3 The requested data is therefore accessible via:
 - <http://comtrade.un.org/api/get?max=10000&type=C&freq=A&px=S2&ps=2021&r=all&p=156&rg=2&cc=AG2>

Step 2: Understand the API call

Table: API explanation

| Parameter | Range | Category | In simple words |
|-----------|------------------------|-----------------------|---|
| max=502 | [1, 10000] | Limit | Data download limits |
| type=C | [C,S] | Data type | C is goods; S is Services |
| freq=A | [A,M] | Obs. frequency | A is annual frequency; M is monthly |
| px=S2 | S[i], H[i], i = [1,4] | Classification system | Product code classification |
| ps=2021 | [1962,2022] | Time period | Year |
| r=all | All, UN codes | Reporting area | All is all countries (not aggregated) |
| p=156 | All, UN codes | Partner(s) | 156 is China's UN numerical country code |
| rg=2 | [1,2,all] | Trade flow | 2 is export from reporting area to partner, 1 is import |
| cc=AG2 | [TOTAL, AG1, ..., AG6] | Classification code | AG2 is all of the 2-digit codes |

`/api/get? [Parameter1] & [Parameter2] &`
`{ [Parameter] = [choice] }`

Step 3 Check the API link

<http://comtrade.un.org/api/get?max=10000&type=C&freq=A&px=S2&ps=2021&r=all&p=156&rg=2&cc=AG2> should show a JSON file:

The screenshot shows a web browser displaying a JSON response from the URL `https://comtrade.un.org/api/get?max=10000&type=C&freq=A&px=S2&ps=2021&r=all&p=156&rg=2&cc=AG2`. The JSON response is a large object with a `dataset` array. A red box highlights the `aggrLevel` field in the first element of the array, which is set to 2. An 'Extensions' menu is open over the page, with 'JSON Formatter' highlighted.

```
{
  "validation": {
    "status": {
      "name": "Ok",
      "value": 0,
      "category": 0,
      "description": "",
      "helpUrl": "For more reference visit http://comtrade.un.org/data/dev/portal/"
    },
    "started": "2022-08-04T10:53:13.3036389+02:00",
    "finished": "2022-08-04T10:53:13.7656126+02:00",
    "durationSeconds": 0.4619737,
    "datasetTimer": {
      "started": "2022-08-04T10:53:13.7656126+02:00",
      "finished": "2022-08-04T10:53:13.7656126+02:00",
      "durationSeconds": 0.4619737
    },
    "dataset": [
      {
        "pfCode": "S2",
        "yr": 2021,
        "period": 2021,
        "periodDesc": "2021",
        "aggrLevel": 2,
        "IsLeaf": 0,
        "rgCode": 2,
        "rgDesc": "Export",
        "rtCode": 36,
        "rtTitle": "Australia",
        "ptCode": 156,
        "ptTitle": "China",
        "ptISO": "CHN",
        "animals chiefly for food": 1,
        "qtCode": 1,
        "qtDesc": "No",
        "Quantity": null,
        "qtAltCode": null,
        "qtAltDesc": "",
        "TradeQuantity": 0,
        "AltQuantity": null,
        "NetWeight": 0,
        "GrossWeight": null,
        "TradeValue": 221606685,
        "CIFValue": null,
        "FOBValue": null,
        "estCode": 4,
        {"pfCode": "S2", "yr": 2021, "period": 2021, "periodDesc": "2021", "aggrLevel": 2, "IsLeaf": 0, "rgCode": 2, "rgDesc": "Export", "rtCode": 56, "rtTitle": "Belgium", "ptCode": 156, "ptTitle": "China", "ptISO": "CHN", "animals chiefly for food": 1, "qtCode": 1, "qtDesc": "No", "Quantity": null, "qtAltCode": null, "qtAltDesc": "", "TradeQuantity": 0, "AltQuantity": null, "NetWeight": 0, "GrossWeight": null, "TradeValue": 3403223, "CIFValue": null, "FOBValue": null, "estCode": 4}, {"pfCode": "S2", "yr": 2021, "period": 2021, "periodDesc": "2021", "aggrLevel": 2, "IsLeaf": 0, "rgCode": 2, "rgDesc": "Export", "rtCode": 76, "rtTitle": "Brazil", "ptCode": 156, "ptTitle": "China", "ptISO": "CHN", "animals chiefly for food": 1, "qtCode": 1, "qtDesc": "No", "Quantity": null, "qtAltCode": null, "qtAltDesc": "", "TradeQuantity": 0, "AltQuantity": null, "NetWeight": 0, "GrossWeight": null, "TradeValue": 551, "CIFValue": null, "FOBValue": null, "estCode": 4}, {"pfCode": "S2", "yr": 2021, "period": 2021, "periodDesc": "2021", "aggrLevel": 2, "IsLeaf": 0, "rgCode": 2, "rgDesc": "Export", "rtCode": 104, "rtTitle": "Myanmar", "ptCode": 156, "ptTitle": "China", "ptISO": "CHN", "animals chiefly for food": 1, "qtCode": 1, "qtDesc": "No", "Quantity": null, "qtAltCode": null, "qtAltDesc": "", "TradeQuantity": 0, "AltQuantity": null, "NetWeight": 0, "GrossWeight": null, "TradeValue": 13500, "CIFValue": null, "FOBValue": null, "estCode": 4}, {"pfCode": "S2", "yr": 2021, "period": 2021, "periodDesc": "2021", "aggrLevel": 2, "IsLeaf": 0, "rgCode": 2, "rgDesc": "Export", "rtCode": 124, "rtTitle": "Canada", "ptCode": 156, "ptTitle": "China", "ptISO": "CHN", "animals chiefly for food": 1, "qtCode": 1, "qtDesc": "No", "Quantity": null, "qtAltCode": null, "qtAltDesc": "", "TradeQuantity": 0, "AltQuantity": null, "NetWeight": null, "GrossWeight": null, "TradeValue": 1559923, "CIFValue": null, "FOBValue": null, "estCode": 0}, {"pfCode": "S2", "yr": 2021, "period": 2021, "periodDesc": "2021", "aggrLevel": 2, "IsLeaf": 0, "rgCode": 2, "rgDesc": "Export", "rtCode": 152, "rtTitle": "Chile", "ptCode": 156, "ptTitle": "China", "ptISO": "CHN", "animals chiefly for food": 1, "qtCode": 1, "qtDesc": "No", "Quantity": null, "qtAltCode": null, "qtAltDesc": "", "TradeQuantity": 0, "AltQuantity": null, "NetWeight": 0, "GrossWeight": null, "TradeValue": 32262991, "CIFValue": null, "FOBValue": null, "estCode": 4}, {"pfCode": "S2", "yr": 2021, "period": 2021, "periodDesc": "2021", "aggrLevel": 2, "IsLeaf": 0, "rgCode": 2, "rgDesc": "Export", "rtCode": 208, "rtTitle": "Denmark", "ptCode": 156, "ptTitle": "China", "ptISO": "CHN", "animals chiefly for food": 1, "qtCode": 1, "qtDesc": "No"}
    ]
  }
}
```

Step 4.1 Set up the destination folder:

- **Task_folder**

- code
- output

```
local filepath = "/Users/.../blogpost1_api_comtrade" // adjust to yours
*country i export to china
cd [filepath]/output/i_X_CHN
```

```
python:
```

```
[python code]
```

```
end
```

Step 4.2 Initiate Python in Stata do-file

```
local filepath = "/Users/.../blogpost1_api_comtrade" // adjust to yours
*country i export to china
cd `filepath'/output/i_X_CHN
```

```
python:
```

```
[python code]
```

```
end
```

Step 4.3 Add necessary Python packages

```
python:
import json
import numpy      as np
import pandas     as pd
import requests

url      =
↪ f'http://comtrade.un.org/api/get?max=10000&type=C&freq=A&px=S2&ps=2021&r=all&p=156&rg=2&cc=AG2'

result   = requests.get(url).json()
if 'dataset' in result:
    df     = pd.DataFrame(result['dataset'])
    df     = df.replace({None: np.nan})
    df.columns= [i[:32] for i in df.columns]
    df.to_stata(f'i_X_China_2021.dta')

end

use ./i_X_China_2021, clear
```

Step 4.4 API link and JSON file

```
python:
import json
import numpy          as np
import pandas        as pd
import requests

url =
↪ f'http://comtrade.un.org/api/get?max=10000&type=C&freq=A&px=S2&ps=2021&r=all&p=156&rg=2&cc=AG2'

result = requests.get(url).json()
if 'dataset' in result:
    df = pd.DataFrame(result['dataset'])
    df = df.replace({None: np.nan})
    df.columns= [i[:32] for i in df.columns]
    df.to_stata(f'i_X_China_2021.dta')
end

use ./i_X_China_2021, clear
```

Step 4.5 Fish the data

5) Recall that we found our data of interest from “dataset” : in Step 3

```
python:
import json
import numpy      as np
import pandas     as pd
import requests

url              =
↳ f'http://comtrade.un.org/api/get?max=10000&type=C&freq=A&px=S2&ps=2021&r=all&p=156&rg=2&cc=AG2'

result          = requests.get(url).json()
if 'dataset' in result:
    df            = pd.DataFrame(result['dataset'])
    df            = df.replace({None: np.nan})
    df.columns   = [i[:32] for i in df.columns]
    df.to_stata(f'i_X_China_2021.dta')

end

use ./i_X_China_2021, clear
```

Step 4.6 Store the data in Stata format (.dta)

```
python:
import json
import numpy          as np
import pandas        as pd
import requests

url      =
↪ f'http://comtrade.un.org/api/get?max=10000&type=C&freq=A&px=S2&ps=2021&r=all&p=156&rg=2&cc=AG2'

result   = requests.get(url).json()
if 'dataset' in result:
    df      = pd.DataFrame(result['dataset'])
    df      = df.replace({None: np.nan})
    df.columns= [i[:32] for i in df.columns]
    df.to_stata(f'i_X_China_2021.dta')
end

use ./i_X_China_2021, clear
```


Step 4.7 Check the result

```
python:
import json
import numpy      as np
import pandas     as pd
import requests

url      =
↪ f'http://comtrade.un.org/api/get?max=10000&type=C&freq=A&px=S2&ps=2021&r=all&p=156&rg=2&cc=AG2'

result   = requests.get(url).json()
if 'dataset' in result:
    df     = pd.DataFrame(result['dataset'])
    df     = df.replace({None: np.nan})
    df.columns= [i[:32] for i in df.columns]
    df.to_stata(f'i_X_China_2021.dta')

end

use ./i_X_China_2021, clear
```

Step 4 Test run result

Data Editor (Browse) — i_X_China_2021.dta

Stata/SE 17.0 — i_X_China_2021.dta

Data Editor (Browse) — i_X_China_2021.dta

index[1] 0

| | index | pfCode | yr | period | periodDesc | aggrLevel | IsLe |
|----|-------|--------|------|--------|------------|-----------|------|
| 1 | 0 | S2 | 2021 | 2021 | 2021 | 2 | |
| 2 | 1 | S2 | 2021 | 2021 | 2021 | 2 | |
| 3 | 2 | S2 | 2021 | 2021 | 2021 | 2 | |
| 4 | 3 | S2 | 2021 | 2021 | 2021 | 2 | |
| 5 | 4 | S2 | 2021 | 2021 | 2021 | 2 | |
| 6 | 5 | S2 | 2021 | 2021 | 2021 | 2 | |
| 7 | 6 | S2 | 2021 | 2021 | 2021 | 2 | |
| 8 | 7 | S2 | 2021 | 2021 | 2021 | 2 | |
| 9 | 8 | S2 | 2021 | 2021 | 2021 | 2 | |
| 10 | 9 | S2 | 2021 | 2021 | 2021 | 2 | |
| 11 | 10 | S2 | 2021 | 2021 | 2021 | 2 | |
| 12 | 11 | S2 | 2021 | 2021 | 2021 | 2 | |
| 13 | 12 | S2 | 2021 | 2021 | 2021 | 2 | |
| 14 | 13 | S2 | 2021 | 2021 | 2021 | 2 | |
| 15 | 14 | S2 | 2021 | 2021 | 2021 | 2 | |
| 16 | 15 | S2 | 2021 | 2021 | 2021 | 2 | |
| 17 | 16 | S2 | 2021 | 2021 | 2021 | 2 | |
| 18 | 17 | S2 | 2021 | 2021 | 2021 | 2 | |
| 19 | 18 | S2 | 2021 | 2021 | 2021 | 2 | |
| 20 | 19 | S2 | 2021 | 2021 | 2021 | 2 | |
| 21 | 20 | S2 | 2021 | 2021 | 2021 | 2 | |

How about looping it over years?

python:

```
url =  
↳ f'http://comtrade.un.org/api/get?max=10000&type=C&freq=A&px=S2&ps=2021&r=all&p=156&rg=2&cc=AG2'
```

end

{[Parameter1] = [choice1] } & {[Parameter2] = [choice2] } & ...

Step 5.1 Compartmentalize the API link

```
python:
def Comtrade_Scraper (ps: int,
                      type: str= 'C',
                      freq: str= 'A',
                      px : str= 'S2',
                      r  : str= 'all',
                      p  : int= 156,
                      rg : int= 2,
                      cc : str= 'AG2'):

    base      = 'https://comtrade.un.org/api/get?max=10000'
    url       = f'{base}&type={type}&freq={freq}&px={px}&ps={ps}&r={r}&p={p}&rg={rg}&cc={cc}'
    result    = requests.get(url).json()
    if 'dataset' in result:
        df      = pd.DataFrame(result['dataset'])
        df      = df.replace({None: np.nan})
        df.columns= [i[:32] for i in df.columns]
        df.to_stata(f'i_X_China_{ps}.dta')
        return df
for i in range(2000,2022): Comtrade_Scraper(i)
end
```

Step 5.2 Assemble the API link

```
python:
def Comtrade_Scraper (ps: int,
                      type: str= 'C',
                      freq: str= 'A',
                      px : str= 'S2',
                      r  : str= 'all',
                      p  : int= 156,
                      rg : int= 2,
                      cc : str= 'AG2'):

    base      = 'https://comtrade.un.org/api/get?max=10000'
    url       = f'{base}&type={type}&freq={freq}&px={px}&ps={ps}&r={r}&p={p}&rg={rg}&cc={cc}'
    result    = requests.get(url).json()
    if 'dataset' in result:
        df      = pd.DataFrame(result['dataset'])
        df      = df.replace({None: np.nan})
        df.columns= [i[:32] for i in df.columns]
        df.to_stata(f'i_X_China_{ps}.dta')
        return df
    for i in range(2000,2022): Comtrade_Scraper(i)
end
```

Step 5.3 Introduce the loop

```
python:
def Comtrade_Scraper (ps: int,
                      type: str= 'C',
                      freq: str= 'A',
                      px : str= 'S2',
                      r  : str= 'all',
                      p  : int= 156,
                      rg : int= 2,
                      cc : str= 'AG2'):

    base      = 'https://comtrade.un.org/api/get?max=10000'
    url       = f'{base}&type={type}&freq={freq}&px={px}&ps={ps}&r={r}&p={p}&rg={rg}&cc={cc}'
    result    = requests.get(url).json()
    if 'dataset' in result:
        df      = pd.DataFrame(result['dataset'])
        df      = df.replace({None: np.nan})
        df.columns= [i[:32] for i in df.columns]
        df.to_stata(f'i_X_China_{ps}.dta')
        return df
for i in range(2000,2022): Comtrade_Scraper(i)
end
```

Step 5.4 Result

| Name | Date Modified | Size | Kind |
|--------------------|----------------|--------|-----------------|
| code | Today at 14:01 | -- | Folder |
| images | Today at 14:11 | -- | Folder |
| output | Today at 12:02 | -- | Folder |
| i_X_CHN | Today at 14:11 | -- | Folder |
| i_X_China_2000.dta | Today at 14:02 | 939 KB | Stata Data File |
| i_X_China_2001.dta | Today at 14:03 | 996 KB | Stata Data File |
| i_X_China_2002.dta | Today at 14:03 | 1 MB | Stata Data File |
| i_X_China_2003.dta | Today at 14:04 | 1.1 MB | Stata Data File |
| i_X_China_2004.dta | Today at 14:04 | 1.1 MB | Stata Data File |
| i_X_China_2005.dta | Today at 14:05 | 1.2 MB | Stata Data File |
| i_X_China_2006.dta | Today at 14:06 | 1.3 MB | Stata Data File |
| i_X_China_2007.dta | Today at 14:06 | 1.3 MB | Stata Data File |
| i_X_China_2008.dta | Today at 14:06 | 1.3 MB | Stata Data File |
| i_X_China_2009.dta | Today at 14:07 | 1.4 MB | Stata Data File |
| i_X_China_2010.dta | Today at 14:07 | 1.5 MB | Stata Data File |
| i_X_China_2011.dta | Today at 14:07 | 1.5 MB | Stata Data File |
| i_X_China_2012.dta | Today at 14:08 | 1.5 MB | Stata Data File |
| i_X_China_2013.dta | Today at 14:08 | 1.5 MB | Stata Data File |
| i_X_China_2014.dta | Today at 14:08 | 1.5 MB | Stata Data File |
| i_X_China_2015.dta | Today at 14:09 | 1.5 MB | Stata Data File |
| i_X_China_2016.dta | Today at 14:09 | 1.6 MB | Stata Data File |
| i_X_China_2017.dta | Today at 14:09 | 1.5 MB | Stata Data File |
| i_X_China_2018.dta | Today at 14:10 | 1.5 MB | Stata Data File |
| i_X_China_2019.dta | Today at 14:10 | 1.5 MB | Stata Data File |
| i_X_China_2020.dta | Today at 14:10 | 1.4 MB | Stata Data File |
| i_X_China_2021.dta | Today at 14:10 | 1 MB | Stata Data File |

Step 5.5 Final touch

6) Final touch

```
di "`c(pwd)'" // Display path to current folder
local files : dir "`c(pwd)'" files "*.dta"
foreach x of local files {
    di "`x'" // Display file name
    append using `x'
}
save ./i_X_China_2000_2021.dta, replace
```

Some final notes

- There is an existing user-written package [comtrade](#) by [Jan Ditzen](#).
 - Last updated on June 2020
- Comtrade recently published a [new API portal](#)
- Data limit remains a challenge, 10'000 for guest users
- There is often glitch in at least 1 file
- Possible to do two-way loop, e.g. over partner country and over time
- There is potential to rewrite into a more flexible structure in order to accommodate multiple data platforms

Credit and reference

Credits to [Satyam Anand](#) (Georgetown University) and [Bernhard Bieri](#) (World Bank) whom introduced the API magic to me.

More resources:

- [Setting up Stata to use Python](#) by Chuck Huber
- [Stata-Python integration blog post](#) by Chuck Huber
- [Comtrade API documentation](#)

Full blog post on <https://www.stevekwong.com/blog/apicomtrade>