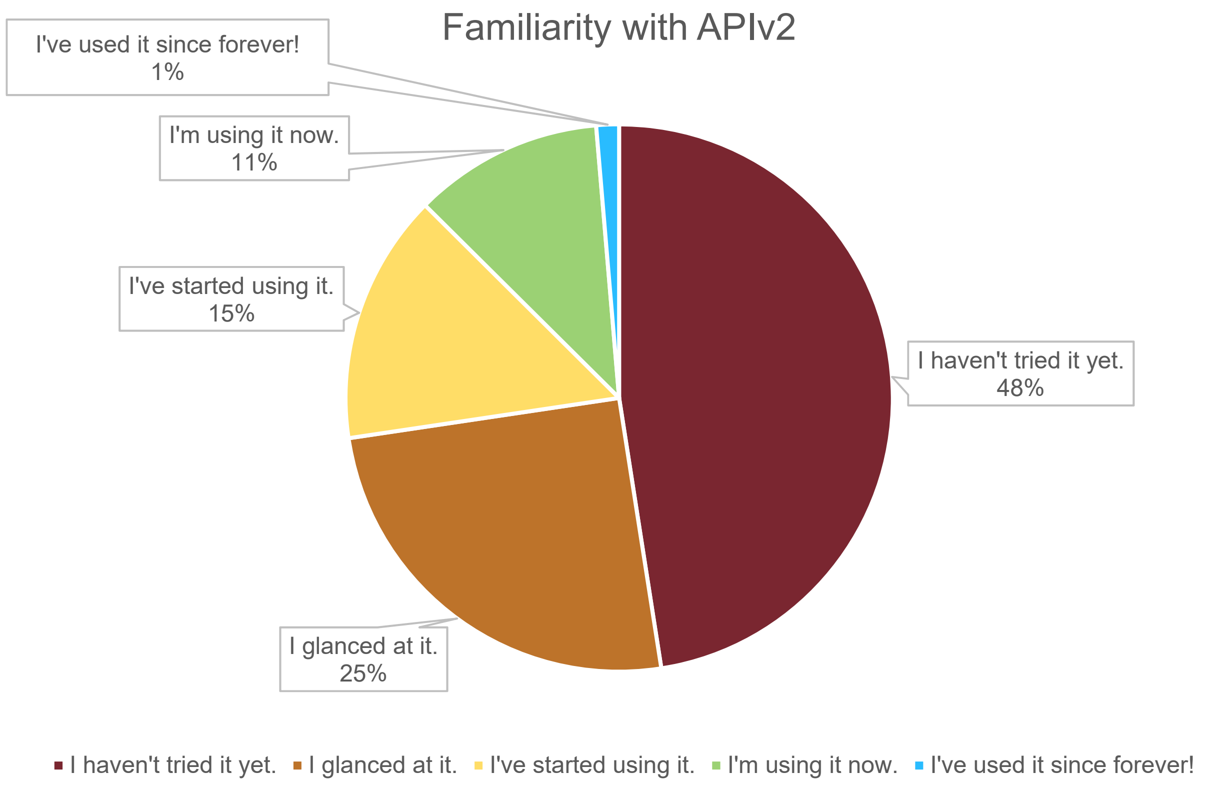
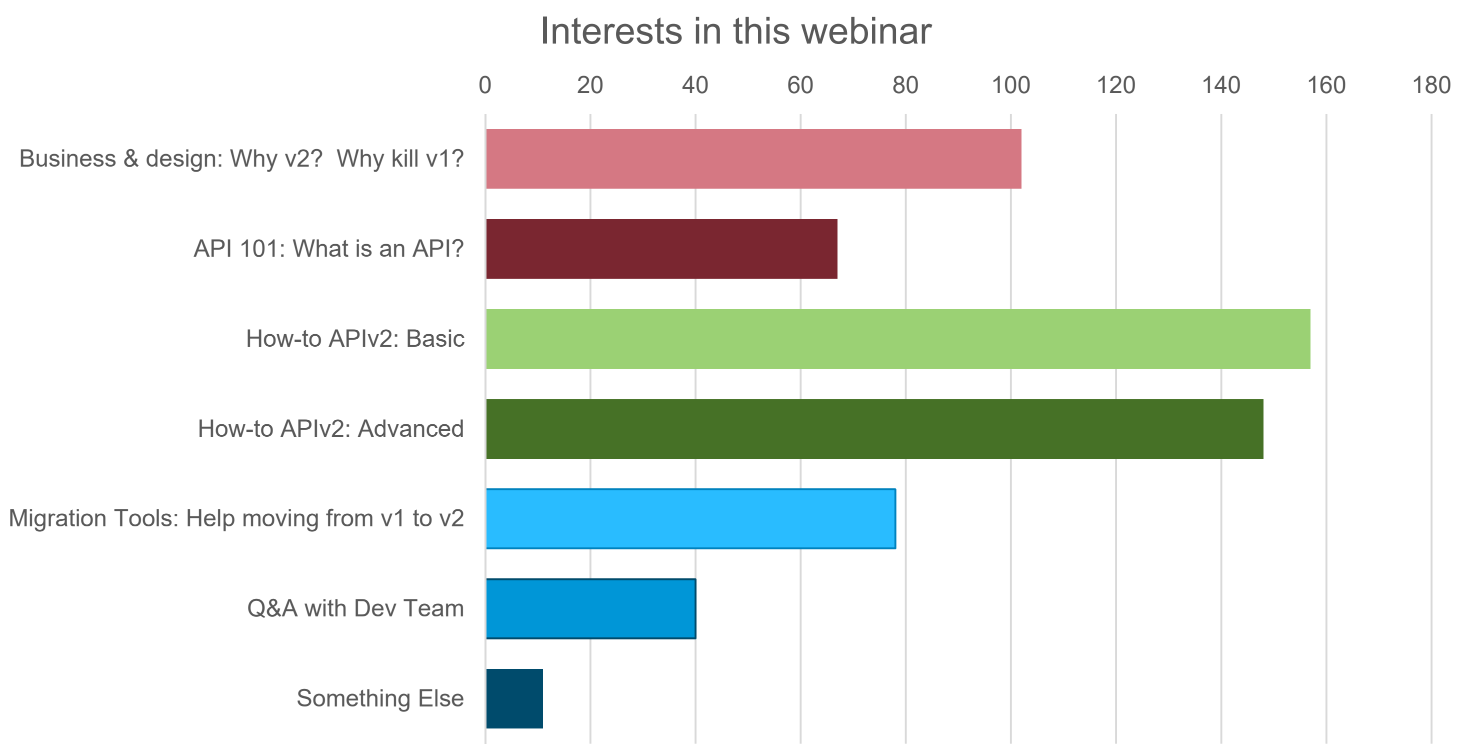
APIv2 Webinar

EIA's API Community
Steve Luminati, Lead Web Project Manager
January 11, 2023 | virtual

Registration results



Registration results



Today’s agenda

- We have a diverse audience in experience.
- We have tailored this webinar to your feedback.
- Do you have questions?
  - We received your prior questions. Thank you!
  - You may submit in real time via your WebEx controls.
- Bear with us:
  - Experts, hang tight through the intro sections.
  - This will be a live demo. Luminati may fumble typing.
- This session is being recorded.

API 101

“The most elementary and valuable statement in science, the beginning of wisdom is: ‘I do not know.’”

-Lt Cmdr Data, USS Enterprise

Layman’s API definition

- Application Programming Interface
  (Not the American Petroleum Institute)

- An interface that exposes some of a system’s internal data and functions so that other developers and systems may make use of it.

- Today’s complex software – phones, desktops, aircraft, commerce, games… all use APIs to rely on others’ work.

- EIA’s API is *read-only* and is *data-only.*

Writing a GPS app without vs. with an API

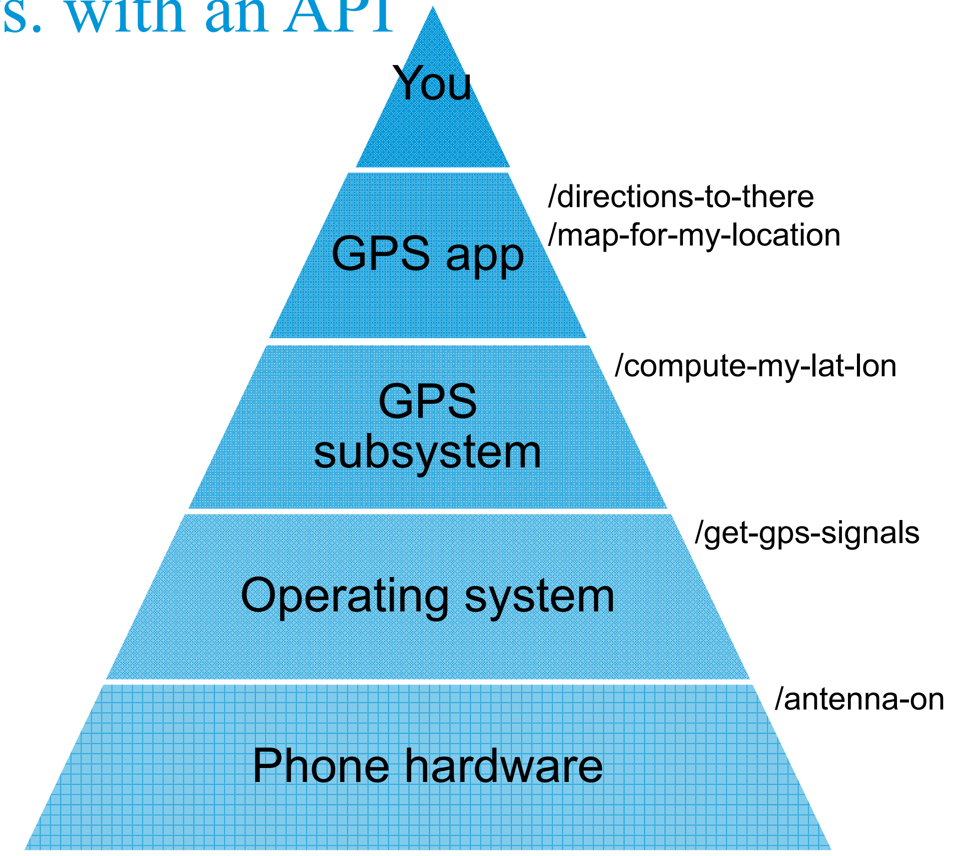
```
;This is assembler.  Good times.

S0: LDA $clues
    ADC $ideas
    MOV R0, 0
    BEQ R0, S2
    JMP S2

S1: MOV 48, R1
    MOV R1, #screen
    MOV 49, R1
    MOV R1, #screen

S2: NOOP


 ;and millions of lines more
 ;everything done yourself
```



API: An alternative to www.eia.gov

- Why would someone ***not*** want to use eia.gov?
  - Personal preference: They don't like the site, our table browsers, or Tableau visualizations
  - The UX doesn't let them manipulate the data the way that they want
  - They need a data *quick hit*; they know exactly what they want and just need the raw numbers
  - They want to check our data with high frequency
  - Using *screen scrapers* to harvest site content is expensive both for user and EIA
  - PDF, CSV *exports* pre-bake data in inflexible formats

    (Our Excel/Sheet add-ins call the API! Technically, those aren't exports.)

- For your use case: No web scraping! No clicking through data browsers!

An example of *stateless*, part of a RESTful API

```
YOU: Tell me a joke.

RESTFUL API: Why did the chicken cross the road?

YOU: Tell me the punchline.

RESTFUL API: … the punchline to what?
```

Business and Design

“Normal people… believe that if it ain’t broke, don’t fix it. Engineers believe that if it ain’t broke, it doesn’t have enough features yet.”

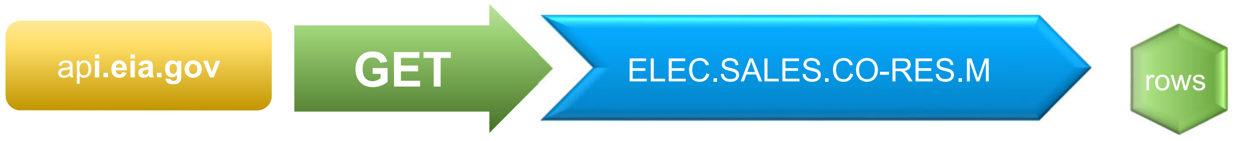
- Douglas Adams

History of APIv1

- APIv1 launched in 2012 and has grown tremendously over 10 years
  - Contains over 1.8 million series
  - Almost 200,000 users
- In that 10 years, Internet best practices have changed
  - General Internet use of public APIs gain acceptance over scraping
  - Automatic discovery becomes standard
  - Interlinking between APIs becomes commonplace
  - RESTful (Representational State Transfer) expectations strengthen
  - API proxies are needed to guard against abuse
- EIA has evolved from a *paper-first* toward a *data-first* operation

An APIv1 call (semi-RESTful)

- A ***target:*** API host
- A ***verb:*** HTTP Method: GET, POST, etc
- A ***series name***: Uniquely, IDs specific, data facets pre-baked
- Optional ***modifiers*** (row count, start and end)
- Data is pre-baked into date-time series



```
https://api.eia.gov/series/?series_id=ELEC.SALES.CO-RES.M&api_key=xxxxx
```

Why APIv2?

- Modern APIs are *machine-discoverable, human-readable,* and *self-describing*
- *Machine-discoverable:* An automated program can traverse our data tree and determine by itself what's available. No data browser required.
- *Human-readable:* A human being can read and understand an API query and its response.
- *Self-describing*: The API itself explains how to use it, what data, facets, and parameters are available.

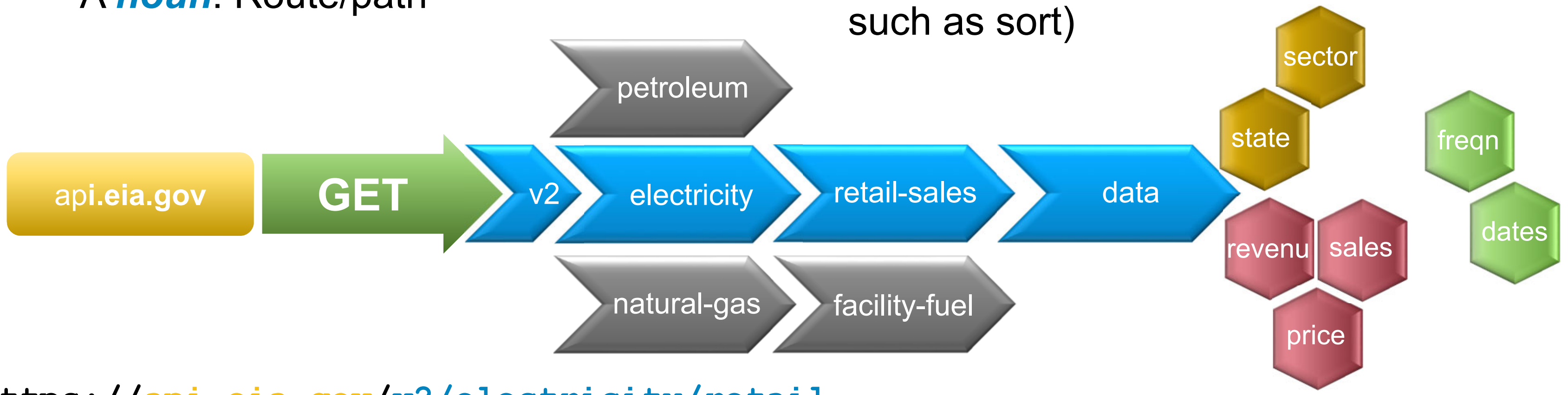
Why APIv2?

- No *secret knowledge* required (avoid use of cryptic *seriesID*)
- Data quality: v2 reaches back to authoritative databases approved by energy analysts, it doesn't contain business logic (data transformations)
- Publishing speed: APIv1 required a lot of manual *hand cranking*
- Improved metadata: Consistent presentation, even when metadata itself isn't
- Security and reliability: Stronger analytics for future products, isolates bad actors
- Creature comforts: Errors and warnings in returns

APIv2: RESTful APIs and routes

Data can be discovered, selected, and interacted with

- A ***target:*** base URI, or API host
- A ***verb:*** HTTP Method: GET, POST, etc
- A ***noun***: Route/path
- Optional ***adverbs*** (facets that specify specific data)
- Optional ***modifiers*** (parameters/filters such as sort)



```
https://api.eia.gov/v2/electricity/retail-
sales/data?data[]=price&facets[sectorid][]=RES&facets[stateid][]=CO&frequency=
monthly&api_key=xxxxxx
```

Why APIv1 will sunset: respect to the taxpayer

- Data quality:
  - v1 and v2 use different data preparation mechanics
  - APIv1 data would not be updated and eventually age-out
- Implementation cost:
  - During development, our hosting platform changed, resulting in a cascade of dependencies
  - APIv1 would be expensive to refit to the new environment
  - Why spend money to rewrite APIv1 if its data won’t be updated?
- Ownership cost:
  - EIA would have to maintain two hosted systems, at double the cost and footprint
  - New customer confusion: Use v1 or v2?

API roadmap

- Version 2.0.1 *Prometheus*: community release (public beta)
  - Date: February 2022
  - Give early-adopters a head start on APIv2
- Version 2.0.2 *Sisyphus*: public launch
  - Data verified by EIA's data/energy experts as good
  - Web-based tools released
  - Further minor revisions (2.0.x) included support for XML, performance optimization, and bugfixes

API roadmap

- Version 2.1 *Clementia*: backwards compatibility (we are here)
  - APIv1 translator: converts an APIv1 seriesID into an APIv2 route
  - Improved query builder (bookmarkable URLs)
  - Port interactive web tools (Grid monitor, et al) to use APIv2
  - Restored bulk downloads
  - Added in-return errors and warnings
- Version 2.2 *Demeter*: (exact scope/date TBD)
  - Gathering feedback now!
  - Definitely: More extensive warning and query feedback, ability for EIA non-programmers to place warnings directly inside API returns (data lag, etc)
  - Possibilities: Additional data sets? Further speed up business process for data publishing? Improved database performance? Restore last-updated field? Restore Google add-In?

How-To Basics

“I have not failed.   I have just found ten-thousand ways that won’t work.”

- Thomas Edison

Getting Setup

- Register for a key
- Use the table browser to get warmed up. It will build API calls for you:

  `https://www.eia.gov/opendata/browser/`

- Read the documentation! ☺

  https://www.eia.gov/opendata/documentation.php

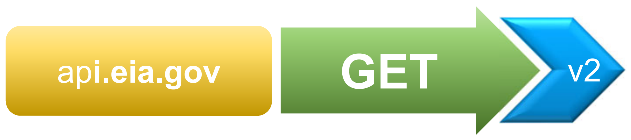
- Now let's go hands on…

  (I'll improv, but this is the same use case as in the documentation, so you can follow along)

Let’s start at the top

(Remember, you’ll use your own API key instead of this one)

```
https://api.eia.gov/v2/?api_key=cbd67337cbc4bed426ef3682226bdab2
```



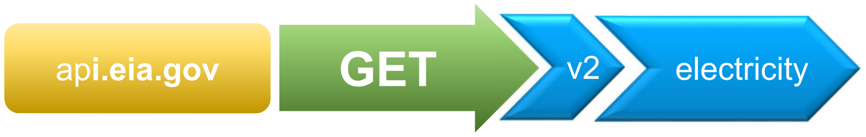
API returns the routes available to you.   This API is *describing itself*.

(You could also put requests in HTTP headers, but for this demo I’ll put everything in the URL)

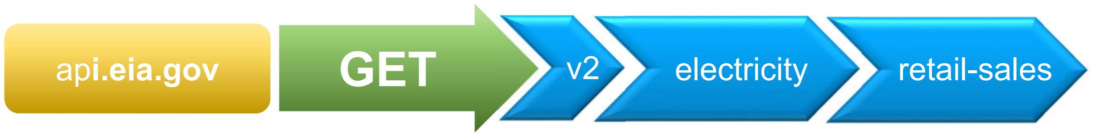
Drilling down

- Let’s explore electricity.

```
https://api.eia.gov/v2/electricity/?api_key=cbd67337cbc4bed426ef3682226bdab
2
```



```
https://api.eia.gov/v2/electricity/retail-
sales/?api_key=cbd67337cbc4bed426ef3682226bdab2
```



- When we get to the last route, API shows the `data[]`object.

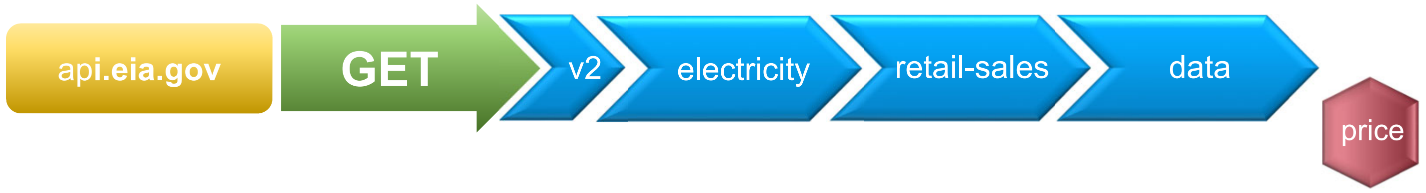
Specifying data

We must end at the `/data` node to get data:

```
https://api.eia.gov/v2/electricity/retail-sales/data
/?api_key=cbd67337cbc4bed426ef3682226bdab2
```

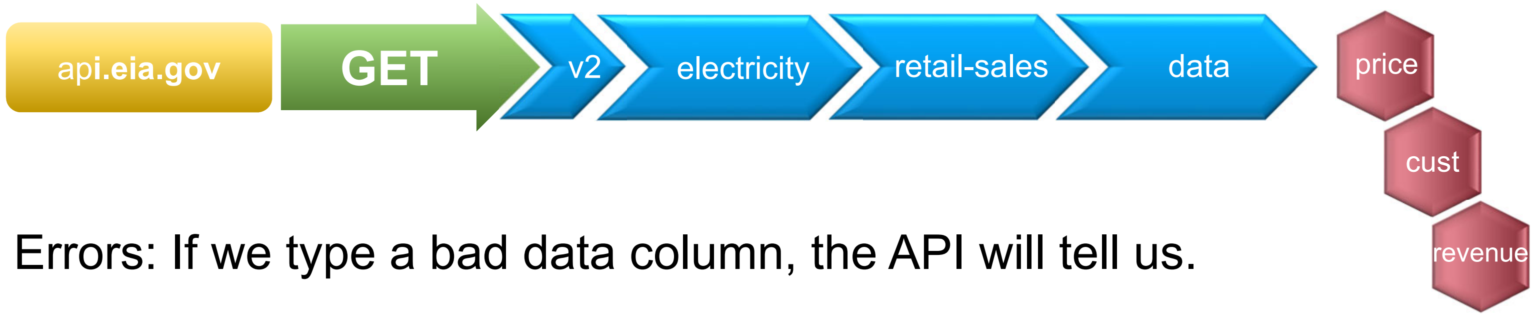
and specify which columns we want returned:

```
https://api.eia.gov/v2/electricity/retail-sales/data?data[]=price
?api_key=cbd67337cbc4bed426ef3682226bdab2
```



Asking for more data columns

```
https://api.eia.gov/v2/electricity/retailsales/data
?data[]=price&data[]=revenue&data[]=customers
&api_key=cbd67337cbc4bed426ef3682226bdab2
```



- Errors: If we type a bad data column, the API will tell us.
- Warnings: If we select too much data, the API will warn us, and return what it can.

How-To Advanced

“Have no fear of perfection – you’ll never reach it.”

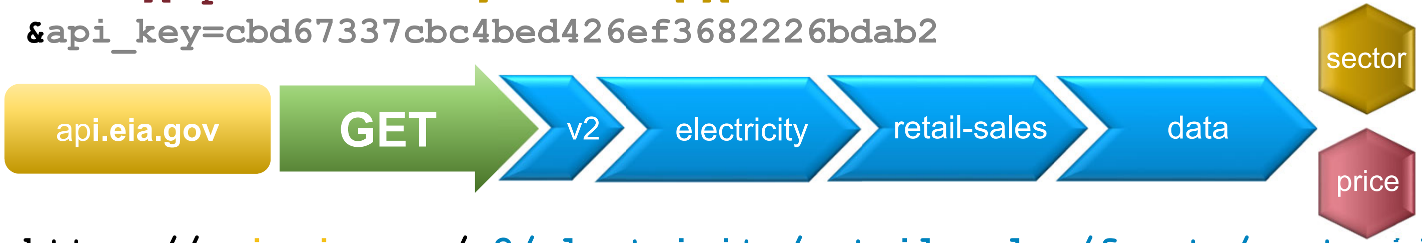
- Salvador Dali

Facets

Remember our last metadata return?

Facets select a dimension or classification of data. Let’s limit our results to the *residential* sector.   (I have removed **`revenue`** and **`customers`** from the **`data[]`** array.)

```
https://api.eia.gov/v2/electricity/retail-sales/data
?data[]=price&facets[sectorid][]=RES
&api_key=cbd67337cbc4bed426ef3682226bdab2
```



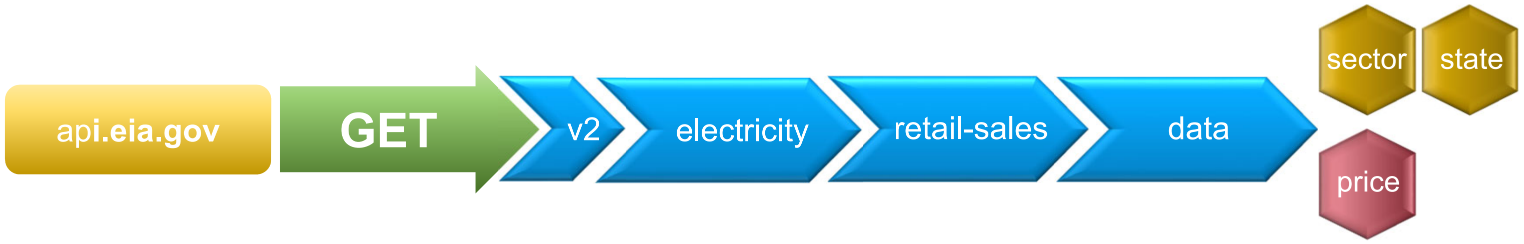
```
https://api.eia.gov/v2/electricity/retail-sales/facets/sectorid
?api_key=cbd67337cbc4bed426ef3682226bdab2
```

Facets

You can add multiple facets, just like multiple data columns.

Let’s further limit our result set to only those from Colorado.

```
https://api.eia.gov/v2/electricity/retail-sales/data
?data[]=price&facets[sectorid][]=RES&facets[stateid][]=CO
&api_key=cbd67337cbc4bed426ef3682226bdab2
```

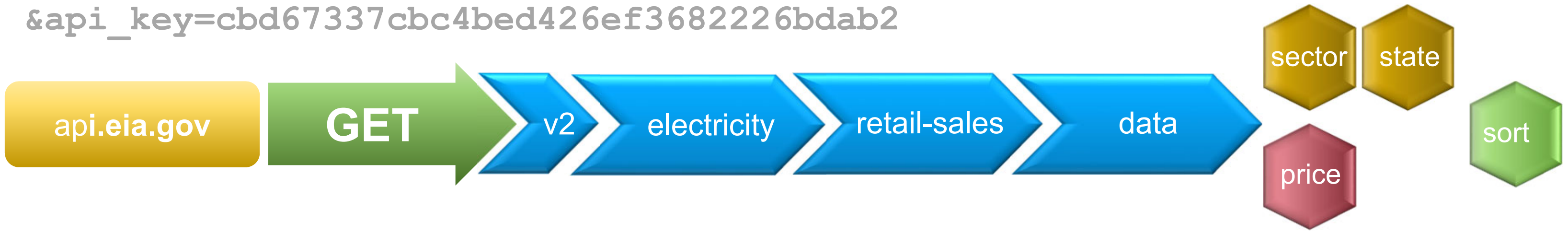


Sorting

Let's make this look like a proper data series by using the parameter **`sort`**.

You don't have to do this, of course, you may prefer your local system to perform the sort (your custom application, off-the-shelf software, etc)

```
https://api.eia.gov/v2/electricity/retail-sales/data
?data[]=price&facets[sectorid][]=RES&facets[stateid][]=CO
&sort[0][column]=period&sort[0][direction]=asc
&api_key=cbd67337cbc4bed426ef3682226bdab2
```

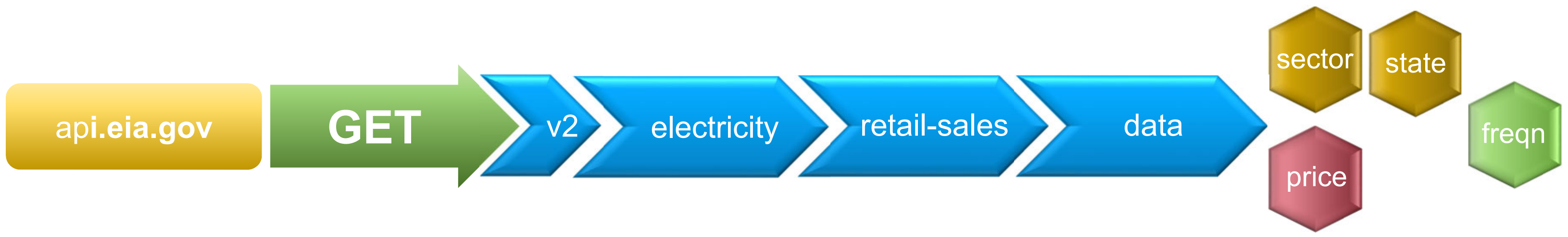


Frequency

Individual date series may express data on an *annual, monthly, quarterly, weekly, daily,* or even *hourly* basis.

A metadata return explains what facets are available. This will vary significantly between data subjects. Metadata is your friend!

```
https://api.eia.gov/v2/electricity/retail-sales/data?
data[]=price&facets[sectorid][]=RES&facets[stateid][]=CO
&frequency=annual&api_key=cbd67337cbc4bed426ef3682226bdab2
```



Frequency defaults and formatting

If you don’t specify a frequency, the API will select a default. This default varies by the data area.

Check the **`dateFormat`** pair to ensure you’re ingesting correctly!

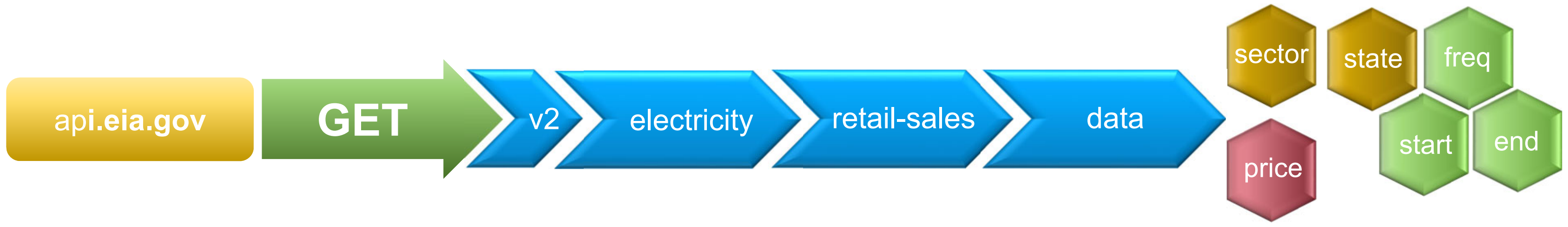
```
response: {
  total: 251
  dateFormat: "YYYY-MM",
  frequency: "monthly",
  …
}
```

Start and end dates

We can still get warnings by >5000 rows.

We did switch to annual data. But if we want monthly, let’s reduce our row count by only asking for one year:

```
https://api.eia.gov/v2/electricity/retail-sales/data
?data[]=price&facets[sectorid][]=RES&facets[stateid][]=CO
&frequency=monthly&start=2007-31-01&end=2008-12-31
&api_key=cbd67337cbc4bed426ef3682226bdab2
```

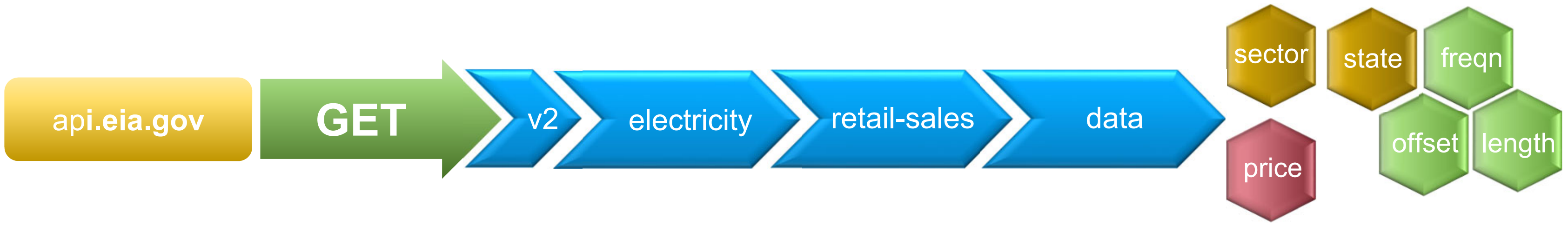


Pagination

Another method of getting around too large a data return is to *paginate*.

We'll ask for a year of data, but this time the 3rd year *in* to the dataset using **`offset`** and **`length`**:

```
https://api.eia.gov/v2/electricity/retail-sales/data
?data[]=price&facets[sectorid][]=RES&facets[stateid][]=CO
&frequency=monthly&offset=24&length=12
&api_key=cbd67337cbc4bed426ef3682226bdab2
```

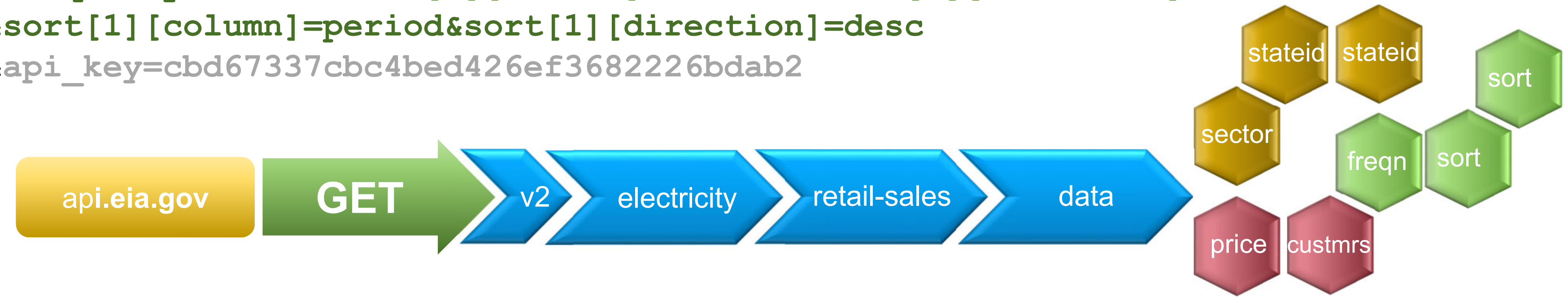


All together now: Who pays more, Virginia or Maryland?

Mix and match what you've learned to build some awesome queries that exactly fit your needs.

(There are other parameters, too, like `output.` Check out the docs!)

```
https://api.eia.gov/v2/electricity/retail-sales/data
?data[]=price&data[]=customers
&facets[sectorid][]=RES&facets[stateid][]=VA&facets[stateid][]=MD
&frequency=annual&sort[0][column]=stateid&sort[0][direction]=desc
&sort[1][column]=period&sort[1][direction]=desc
&api_key=cbd67337cbc4bed426ef3682226bdab2
```



Migration Tools

“If you find yourself in a hole…. stop digging.”

- Will Rogers

APIv1 emulation

- Use the /seriesid/ route with a series v1 ID:

```
https://api.eia.gov/v2/seriesid/ELEC.SALES.CO-RES.A?
api_key=cbd67337cbc4bed426ef3682226bdab2
```

- Data are returned in v2 JSON format

Web-based translator

- Go to this specific URL, enter a v1 seriesID, and see the v2 query that generates it.

  `https://www.eia.gov/opendata/#translate`

- Rendered in the query browser

- Benefit of this method: Using a v2 URL lets you customize the API call

Dev Q&A

“*Expert: a man who makes three correct guesses consecutively.*”

- Laurence J. Peter

For more information

U.S. Energy Information Administration home page | www.eia.gov

EIA’s Open Data | www.eia.gov/opendata

Full API documentation | www.eia.gov/opendata/documentation.php

APIv2 Webinar

EIA's API Community
Steve Luminati, Lead Web Project Manager
January 11th, 2023 | Conducted online