AUSTRALIA - API GUIDE

*Richard Davies*

**Summary**

A guide to using the Australian Bureau of Statistics API. This is a useful process to follow since it shows how tricky APIs can be, but also the value of persevering with them.

**Background and links**

The Australian Bureau of Statistics launched their beta API in March 2022. <https://www.abs.gov.au/about/data-services/application-programming-interfaces-apis/data-api-user-guide>

The base API is this: <https://api.data.abs.gov.au>

Further on we are told that the final aim is a url of the form:

[https://api.data.abs.gov.au/data/{flowRef}/{dataKey}?{queryParameters}](https://api.data.abs.gov.au/data/%7bflowRef%7d/%7bdataKey%7d?%7bqueryParameters%7d)

The challenge is to find the flow ref (quite easy) and the dataKey (extremely tricky).

**Working example:**

Deep within the guide we are given a working example:

<https://api.data.abs.gov.au/data/ALC/1.2.1.4.A>.

To turn this into useable data, we can use this:

<https://api.data.abs.gov.au/data/ALC/1.2.1.4.A?format=jsondata>

The data output in json form is nasty – not formatted in any recognised way and so will need mending using JavaScript.

**CPI example**

From the CPI release we have that the code for CPI is A2325850V. This is from Table 1 here: <https://www.abs.gov.au/statistics/economy/price-indexes-and-inflation/consumer-price-index-australia/latest-release>

This is going to be a red herring, and not related to the API.

For any agency we can get the list of possible data “flows” using:

[https://api.data.abs.gov.au/dataflow/{agencyId}](https://api.data.abs.gov.au/dataflow/%7bagencyId%7d)

This produces a long and messy XML file.

Within this using CTRL-F on “inflation”, we find:

<structure:Dataflow id="CPI" agencyID="ABS" version="1.1.0" isFinal="true">

So no we know the code for inflation is CPI.

With this in hand we look at the data structure.

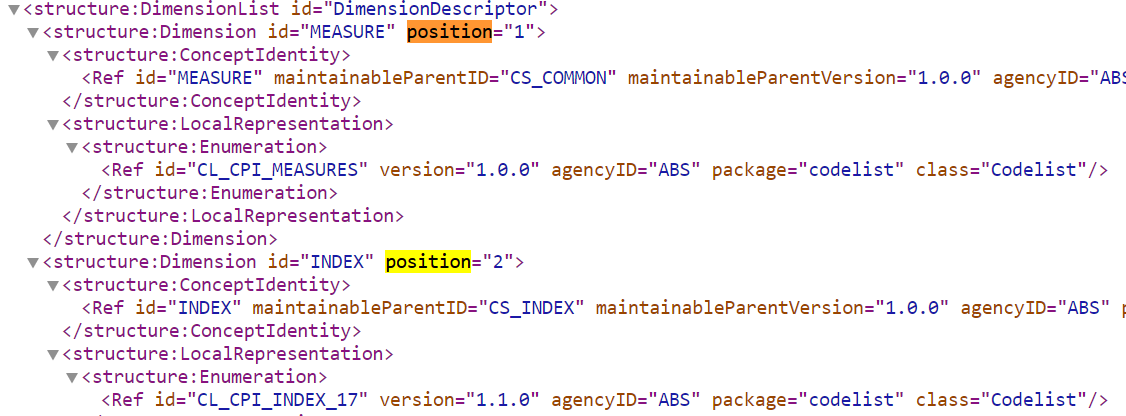
<https://api.data.abs.gov.au/datastructure/ABS/CPI>

Here what we need to search for—using CTRL-F—is the term “position”. This will be key in making our data call. We not only need to know the number of things with a “position” but also what is in them.

For that reason we need the code list:

<https://api.data.abs.gov.au/datastructure/ABS/CPI?references=codelist>

Within this we find this:



Note that what we need is the “Ref id”. Here it helps to make a table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Position | Ref if |  |  |  |
| 1 | CL\_CPI\_MEASURES |  |  |  |
| 2 | CL\_CPI\_INDEX\_17 |  |  |  |
| 3 | CL\_TSET |  |  |  |
| 4 | CL\_CPI\_REGION |  |  |  |
| 5 | CL\_FREQ |  |  |  |
| 6 | TIME\_PERIOD |  |  |  |

From this we know the position of the items we need to make a call on the API. But what do we actually put in? To find this we need to search the code list.

[https://api.data.abs.gov.au/codelist/ABS/{CODES}/1.0.0](https://api.data.abs.gov.au/codelist/ABS/%7bCODES%7d/1.0.0)

In this case:

<https://api.data.abs.gov.au/codelist/ABS/CL_CPI_INDEX_17/1.0.0>

<https://api.data.abs.gov.au/codelist/ABS/CL_TSEST_17/1.0.0>

|  |  |  |  |
| --- | --- | --- | --- |
| Position | Ref if | Code | What? |
| 1 | CL\_CPI\_MEASURES | 3 | Percentage Change from Corresponding Quarter of the Previous Year |
| 2 | CL\_CPI\_INDEX\_17 | 10001 | All Groups CPI |
| 3 | CL\_TSET | 10 | Original |
| 4 | CL\_CPI\_REGION | 50 | Australia |
| 5 | CL\_FREQ | Q | Quarterly |
| 6 | TIME\_PERIOD | XYZ |  |  |

Another way to find this, but less clear, is by using:

<https://api.data.abs.gov.au/datastructure/ABS/CPI?references=children>

Putting this together we have a dataset id of:

**3.10001.10.50.Q**

[https://api.data.abs.gov.au/data/CPI/3.10001.10.50.Q](https://api.data.abs.gov.au/data/CPI/3.10001.10.50.Q )

**Retrieving json**

Now we can add query parameters to the end of it. In particular:

<https://api.data.abs.gov.au/data/CPI/3.10001.10.50.Q?format=jsondata>

As above this is a real mess of a dataset.

The CSV is cleaner:

<https://api.data.abs.gov.au/data/CPI/3.10001.10.50.Q?format=csv>

This will download a file, that is not recorded as CSV to your machine. Open Excel, search for “all files” (since excel will not open it), and then open it, telling Excel that it is comma delimited. Inside the file we see the data:

|  |  |
| --- | --- |
| TIME\_PERIOD | OBS\_VALUE |
| 1949-Q3 | 10.8 |
| 1949-Q4 | 7.9 |
| 1950-Q1 | 7.7 |

**Plotting this in Vega Lite**

In your Vega spec you need to tell vega that this is CSV, since the file does not arrive with a .csv extension. So in the data part of your spec you need:

{ "$schema": "https://vega.github.io/schema/vega-lite/v5.json",

  "data": {

      "url": "https://api.data.abs.gov.au/data/CPI/3.10001.10.50.Q?format=csv",

    "format":{"type": "csv"}},

  "mark": "line",

  "encoding": {

    "x": {"field": "TIME\_PERIOD", "type": "ordinal"},

    "y": {"field": "OBS\_VALUE", "type": "quantitative"}}}

The main problem here is the form of the date variable which is of the form 2008-Q4. This can be dealt with using string and time functions in the Vega spec:

{ "$schema": "https://vega.github.io/schema/vega-lite/v5.json",

  "data": {

        "url": "https://api.data.abs.gov.au/data/CPI/3.10001.10.50.Q?format=csv",

        "format":{"type": "csv"}},

   "transform": [

       {"calculate": "split(datum.TIME\_PERIOD, '-Q')", "as": "temp1"},

       {"calculate": "datum.temp1[0]+'-'+datum.temp1[1]\*3", "as": "temp2"},

       {"calculate": "toDate(datum.temp2)", "as": "date"}

   ],

  "mark": "line",

  "encoding": {

    "x": {"field": "date", "type": "temporal"},

    "y": {"field": "OBS\_VALUE", "type": "quantitative"}}}

**Another example - Unemployment:**

<structure:Dataflow id="ABS\_LABOUR\_ACCT" agencyID="ABS" version="1.0.0" isFinal="true">

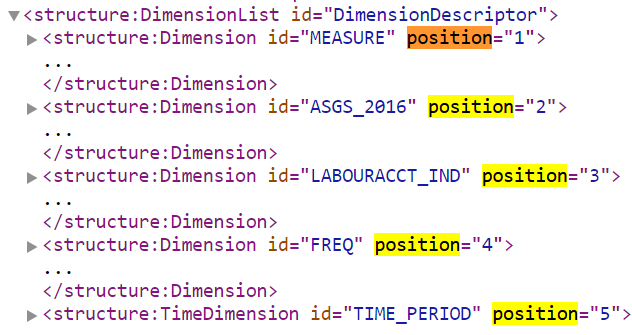
<https://api.data.abs.gov.au/datastructure/ABS/ABS_LABOUR_ACCT>

This gives us the detail of Australia’s “Labour Account”.

Turning to the code lists we get this:

<https://api.data.abs.gov.au/datastructure/ABS/ABS_LABOUR_ACCT?references=codelist>

Looking again we need the things in a specific position:



Under each position we see the Ref-id. Note that this is NOT the same as the id’s you see in the image directly above. i.e. “MEASURE” is not one of them.

|  |  |  |  |
| --- | --- | --- | --- |
| Position |  | Example | What |
| 1 | CL\_LBR\_ACCT\_ITEMS | M9 | Total jobs |
| 2 | CL\_STATE | AUS |  |
| 3 | CL\_ANZSIC\_LA | TOTAL |  |
| 4 | CL\_FREQ | A |  |
| 5 | TIME\_PERIOD |  |  |

Again we use the code lists, e.g.

<https://api.data.abs.gov.au/codelist/ABS/CL_LBR_ACCT_ITEMS/1.0.0>

Putting these together we have our dataKey: **M9.AUS.TOTAL.A**

The final one that works:

<https://api.data.abs.gov.au/data/ABS_LABOUR_ACCT/M9.AUS.TOTAL.A?detail=full&format=jsondata>

The JSON data is horrible in set up:

<https://api.data.abs.gov.au/data/ABS_LABOUR_ACCT/M9.AUS.TOTAL.A?detail=full>

This spec will make a chart:

{"$schema": "https://vega.github.io/schema/vega-lite/v5.json",

  "data": {

    "url": "https://api.data.abs.gov.au/data/ABS\_LABOUR\_ACCT/M9.AUS.TOTAL.A?detail=dataonly&format=csv",

    "format":{"type": "csv"}},

  "mark": "line",

  "encoding": {

    "x": {"field": "TIME\_PERIOD", "type": "temporal"},

    "y": {"field": "OBS\_VALUE", "type": "quantitative"}}

}

**After the pain, the gain**

This all takes much longer than simply finding the Excel sheet and copy and pasting the chart. But what you have built is live, and so will auto update.

We then look back at the codelist:

<https://api.data.abs.gov.au/codelist/ABS/CL_LBR_ACCT_ITEMS/1.0.0>

And we find the following:

* M1 – average income
* M6 – average hours worked per job
* M9 – total jobs
* M10 – Vacancies
* M24 – unemployed
* M26 – underemployed
* M27 – not in the labour force