Contents

Introduction

The Statistics Canada SDMX REST web services provide access to the time series made available on the Statistics Canada's website in a structured form. It complies with the SDMX standard. This service is accessible using an HTTP request.

The time series available through Statistics Canada's Common Output Data Repository (CODR) are presented in a form of Canadian indicators/cubes (short-term, structural, specific, etc.) usually representing a set of series/vectors. An indicator/cube is broken down into as many elementary series as there are possible crossover of variables/dimensions. For example, 50 industries * 13 provinces/territories = 650 different series/vectors in CODR.

The SDMX web service allows access:

- to the values of the series
- structural metadata describing the characteristics of the series
- is free of charge

It is possible to quickly extract data using the CODR cube identifier and vectors. In these cases the user must have previously noted the CODR cube identifier or vector identifier from the Statistics Canada website.

What is SDMX?

The Statistical Data and Metadata Exchange initiative is sponsored by seven institutions (the Bank for International Settlements (BIS), the European Central Bank (ECB), Eurostat, the International Monetary Fund (IMF), the Organisation for Economic Co-operation and Development (OECD), the United Nations (UN) and the World Bank) to foster standards for the exchange of statistical information. The first version of the standard is an ISO standard (ISO/Technical Specification 17369:2005). It offers an information model for the representation of statistical data and metadata, as well as several formats to represent this model (SDMX-EDI, SDMX-JSON, SDMX-CSV and several SDMX-ML formats). It also proposes a standard way of implementing web services, including the use of registries.

The SDMX information model in a nutshell

The list below tells you everything you need to know about the SDMX information model in order for us to start developing an application based on the SDMX standard:

- **Descriptor concepts**: In order to make sense of some statistical data, we need to know the concepts associated with them. For example, on its own the figure 1.3312 is pretty meaningless, but if we know that this is an exchange rate for the CDN dollar against the US dollar on November 19th, 2019, it starts to make more sense.
- **Packaging structure**: Statistical data can be grouped together at the following levels: the observation level (the measurement of some phenomenon); the series level (the measurement of some phenomenon over time, usually at regular intervals); the group level (a group of series – a well-known example being the sibling group, a set of series which are identical, except for the fact that they are measured with different frequencies); and the dataset level (made up of several groups, to cover a specific statistical domain for instance). The descriptor concepts mentioned in point 1 can be attached at various levels in this hierarchy.
- **Dimensions and attributes**: There are two types of descriptor concepts: dimensions, which both identify and describe the data, and attributes, which are purely descriptive.

- **Keys**: Dimensions are grouped into keys, which allow the identification of a particular set of data (a series, for example). The key values are attached at the series level and given in a fixed sequence. Conventionally, frequency is the first descriptor concept and the other concepts are assigned an order for that particular dataset. Partial keys can be attached to groups.
- **Code lists**: Every possible value for a dimension is defined in a code list. Each value on that list is given a language-independent abbreviation (code) and a language-specific description. Attributes are represented sometimes by codes, and sometimes by free-text values. Since the purpose of an attribute is solely to describe and not to identify the data, this is not a problem.
- **Data Structure Definitions**: A Data Structure Definition (key family) specifies a set of concepts, which describe and identify a set of data. It tells us which concepts are dimensions (identification and description) and which are attributes (just description), and it gives the attachment level for each of these concepts on the basis of the packaging structure (dataset, group, series or observation), as well as their status (mandatory or conditional). It also specifies which code lists provide possible values for the dimensions and gives possible values for the attributes, either as code lists or free text fields.

The various SDMX-ML formats

SDMX-ML supports various use cases and, therefore, defines several XML formats. For the purpose of this guide, the following two formats will be used:

- **The Structure Definition format** : This format will be used to define the structure (concepts, code lists, dimensions, attributes, etc.) of the key families.
- **The Compact format**: This format will be used to define the data file. It is not a generic format (it is specific to a Data Structure Definition), but it is designed to support validation and is much more compact so as to support the exchange of large datasets.

The SDMX information model is much richer than this limited introduction, however the above should be sufficient to understand the basics of this web service. For additional information, please refer to the SDMX documentation (SDMX.org).

SDMX Data (cube) Web Service

All the data stored in CODR can be retrieved using the query string described below.

```
protocol://wsEntryPoint/resource/flowRef/key?parameters
```

where parameters are defined as such:

startPeriod=value&endPeriod=value&firstNObservations=value&lastNObservations=value&detail=value

Syntax definition

protocol

The web service is available over http and https.

wsEntryPoint

The web service entry point is available at the same location of the sdmx data and metadata entry point.

resource

The resource for queries is dataflow.

flowRef

A reference to the dataflow describing the data that needs to be returned.

The syntax is the identifier of the agency maintaining the dataflow, followed by the identifier of the dataflow, followed by the dataflow version, separated by a ,(comma).

For example: AGENCY_ID, FLOW_ID, VERSION

If the parameter contains only one of these 3 elements, it is considered to be the identifier of the dataflow. The value for the identifier of the agency maintaining the dataflow will default to all, while the value for the dataflow version will default to latest.

If the string contains only two of these 3 elements, they are considered to be the identifier of the agency maintaining the dataflow and the identifier of the dataflow. The value for the dataflow version will default to latest.

key

The combination of dimensions allows statistical data to be uniquely identified. Such a combination is known as a series key in SDMX and this is what is needed in the key parameter.

Let's say for example that exchange rates can be uniquely identified by the following:

- the frequency at which they are measured (e.g.: on a daily basis - code D),
- the currency being measured (e.g.: US dollar - code USD),
- the currency against which a currency is being measured (e.g.: Euro - code EUR),
- the type of exchange rates (Foreign exchange reference rates - code SP00) and
- the series variation (such as average or standardized measure for given frequency, code A).

In order to build the series key, you need to take the value for each of the dimensions (in the order in which the dimensions are defined in the DSD) and separate them with a .(dot). The series key for the example above therefore becomes: D.USD.EUR.SP00.A

Wildcarding is supported by omitting the value for the dimension to be wildcarded. For example, the following series key can be used to retrieve the data for all daily currencies against the euro: D..EUR.SP00.A

The OR operator is supported using the + (plus) character. For example, the following key can be used to retrieve the exchange rates against the euro for both the US dollar and the Japanese Yen: D.USD+JPY.EUR.SP00.A

You can of course combine wildcarding and the OR operator. For example, the following key can be used to retrieve daily or monthly exchange rates of any currency against the euro: D+M..EUR.SP00.A

startPeriod & endPeriod

It is possible to define a date range for which observations should be returned by using the startPeriod and/or endPeriod parameters. The values should be given according to the syntax defined in ISO 8601 or as SDMX reporting periods. The format will vary depending on the frequency.

The supported formats are:

- YYYY for annual data (e.g.: 2013).
- YYYY-MM for monthly data (e.g.: 2013-01).
- YYYY-MM-DD for daily data (e.g.: 2013-01-01).

Detail

Using the detail parameter, it is possible to specify the desired amount of information to be returned by the web service.

Possible options are:

- full: The data (series and observations) and the attributes will be returned. This is the default.

firstNObservations & lastNObservations

Using the firstNObservations and/or lastNObservations parameters, it is possible to specify the maximum number of observations to be returned for each of the matching series, starting from the first observation (firstNObservations) or counting back from the most recent observation (lastNObservations).

Examples

1. Retrieve the data for the series 1.1.1 (Canada / Both sexes / All ages) for the 17100005 dataflow. Table 17100005 is presented in full details in appendix 1.

https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/data/DF_17100005/1.1.1

Results

SDMX-ML 2.1 **Generic** Data	ex_1_gen.xml
SDMX-ML 2.1 **Structure** Specific Data	ex_1_ssd.xml
SDMX-JSON	ex_1_json.json

2. Retrieve the data for the series 1.2+3.1 (Canada / Male & Female / All ages) for the 17100005 dataflow. Table 17100005 is presented in full details in appendix 1.

https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/data/DF_17100005/1.2+3.1

Results

SDMX-ML 2.1 **Generic** Data	ex_2_gen.xml
SDMX-ML 2.1 **Structure** Specific Data	ex_2_ssd.xml
SDMX-JSON	ex_2_json.json

3. Retrieve the data for the series .1.138 (all geographies / Both sexes / 100 years and over) for the 17100005 dataflow and for the reference years 2015 & 2016. Table 17100005 is presented in full details in appendix 1.

https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/data/DF_17100005/.1.138?startPeriod=2015&endPeriod=2016

Results

SDMX-ML 2.1 **Generic** Data	ex_3_gen.xml
SDMX-ML 2.1 **Structure Specific** Data	ex_3_ssd.xml
SDMX-JSON	ex_3_json.json

SDMX Data (vector) Web Service

All the data stored in CODR can be retrieved using the query string described below.

```
protocol://wsEntryPoint/resource/vector?parameters
```

where parameters are defined as such:

startPeriod=value&endPeriod=value&firstNObservations=value&lastNObservations=value&detail=value

Syntax definition

protocol

The web service is available over http and https.

wsEntryPoint

The web service entry point is available at the same location of the sdmx data and metadata entry point.

resource

The resource for vector queries is vector.

vector

The vector allows statistical time series to be identified. Vectors are unique identifiers to a time series of data points. (i.e. v123456) They do not change over time. Given the vector number does not change, users can still use the same vectors as shortcuts to their data points of interest.

startPeriod & endPeriod

It is possible to define a date range for which observations should be returned by using the startPeriod and/or endPeriod parameters. The values should be given according to the syntax defined in ISO 8601 or as SDMX reporting periods. The format will vary depending on the frequency.

The supported formats are:

- YYYY for annual data (e.g.: 2013).
- YYYY-MM for monthly data (e.g.: 2013-01).
- YYYY-MM-DD for daily data (e.g.: 2013-01-01).

Detail

Using the detail parameter, it is possible to specify the desired amount of information to be returned by the web service.

Possible options are:

- full: The data (series and observations) and the attributes will be returned. This is the default.

firstNObservations & lastNObservations

Using the firstNObservations and/or lastNObservations parameters, it is possible to specify the maximum number of observations to be returned for each of the matching series, starting from the first observation (firstNObservations) or counting back from the most recent observation (lastNObservations).

Examples

4. Retrieve the data for the series (vectors) 466670 (Canada / Male / All ages). Table 17100005 is presented in full details in appendix 1.
https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/vector/v466670

Results

SDMX-ML 2.1 **Generic** Data	ex_2_gen.xml
SDMX-ML 2.1 **Structure** Specific Data	ex_2_ssd.xml
SDMX-JSON	ex_2_json.json

SDMX Metadata (structure) Web Service

5. Retrieve the latest version in production of the DSD with id 17100005 data structure. Table 17100005 is presented in full details in appendix 1.
https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/structure/Data_Structure_17100005

Results

SDMX-ML Structure format	ex_11.xml

Content negotiation

Using the HTTP content negotiation mechanism, you can select the representation to be returned and you can also instruct the service to compress the data to be returned.

Format selection

The following data formats are supported by the web service:

- Data formats:

    - SDMX-ML 2.1 Generic Data format: `application/vnd.sdmx.genericdata+xml;version=2.1`. This is the **default** for data queries.
    - SDMX-ML 2.1 Structure Specific Data format: `application/vnd.sdmx.structurespecificdata+xml;version=2.1`
    - SDMX-JSON: `application/vnd.sdmx.data+json;version=1.0.0-wd`
    - CSV: `text/csv`
- Metadata formats:
    - SDMX-ML Structure format: `application/vnd.sdmx.structure+xml;version=2.1`

For additional information about the various SDMX-ML formats, please refer to the SDMX documentation.

Generic mime types (`application/json`, `application/xml`) are also supported but they will always point to the most recent version of the SDMX formats supported by these web services. That version will change in the future, whenever new versions of the various SDMX formats are made available.

Is it therefore **highly** recommended that implementers use one of the specific mime types above rather than a generic one, to avoid issues when new versions of the formats are released.

Output compression

You can also enable data compression using the Accept-Encoding HTTP header field. Compressed messages are typically significantly smaller than uncompressed messages, which can lead to improvements when transferring large amount data over the network.

Status codes

The web service returns the following HTTP status codes.

Code	Status	Description
200	OK	Your query could be successfully processed and the data have been returned.
304	No changes	No changes since the timestamp supplied in the `If-Modified-Since` header.
400	Syntax error	If there is a syntactic or semantic issue with the parameters you supplied, a 400 HTTP status code will be returned.
404	No results found	A 404 HTTP status code will be returned if there are no results matching the query.

Code	Status	Description
406	Not Acceptable	If you ask for a resource representation that we don't support, a 406 HTTP status code will be returned. See the section about content negotiation, to view the supported representations.
500	Internal Server Error	When there is an issue on our side, a 500 HTTP status code will be returned. Feel free to try again later or to contact our support hotline.
501	Not implemented	This web service offers a subset of the functionality offered by the SDMX RESTful web service specification. When you use a feature that we have not yet implemented, a 501 HTTP status code will be returned.
503	Service unavailable	If our web service is temporarily unavailable, a 503 HTTP status code will be returned.

Useful tips

The SDMX Technical Working Group publishes a list of tips and tricks for web service clients, which is well worth reading.

The SDMX Technical Working Group has also published a cheatsheet which summarises, in 2 A4 pages, the main points of the SDMX 2.1 RESTful API.

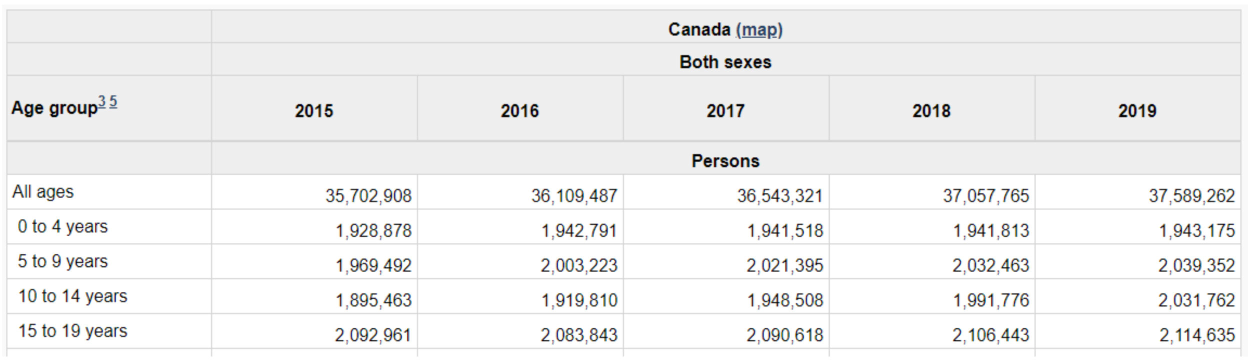
If the documentation does not contain the information you require, or if you have any general comments or feedback regarding our web service, please contact us.

All sample queries in this tutorial can also be executed using command-line tools such as `curl` or `wget`:

```
wget -O data.xml \
--header="Accept:application/vnd.sdmx.structurespecificdata+xml;version=2.1" \
https://sdw-wsrest.ecb.europa.eu/service/data/EXR/M.NOK.EUR.SP00.A
curl -k -o data.xml \
--header "Accept:application/vnd.sdmx.structurespecificdata+xml;version=2.1" \
https://sdw-wsrest.ecb.europa.eu/service/data/EXR/M.NOK.EUR.SP00.A
```

Appendix 1

Table 17100005 - Population estimates on July 1st, by age and sex



	Canada (map)				
	Both sexes				
Age group[3 5]	2015	2016	2017	2018	2019
	Persons				
All ages	35,702,908	36,109,487	36,543,321	37,057,765	37,589,262
0 to 4 years	1,928,878	1,942,791	1,941,518	1,941,813	1,943,175
5 to 9 years	1,969,492	2,003,223	2,021,395	2,032,463	2,039,352
10 to 14 years	1,895,463	1,919,810	1,948,508	1,991,776	2,031,762
15 to 19 years	2,092,961	2,083,843	2,090,618	2,106,443	2,114,635

1st Dimension - Geography

Code	Name_en
1	Canada
2	Newfoundland and Labrador
3	Prince Edward Island
4	Nova Scotia
5	New Brunswick
6	Quebec
7	Ontario
8	Manitoba
9	Saskatchewan
10	Alberta
11	British Columbia
12	Yukon
13	Northwest Territories including Nunavut
14	Northwest Territories
15	Nunavut

2nd Dimension – Sex

Code	Name_en
1	Both sexes
2	Males
3	Females

3rd Dimension – Age group

Code	Name_en

1	All ages
2	0 years
3	1 year
4	2 years
5	3 years
6	4 years
7	0 to 4 years
8	5 years
9	6 years
10	7 years
11	8 years
12	9 years
13	5 to 9 years
14	10 years
15	11 years
16	12 years
17	13 years
18	14 years
19	10 to 14 years
20	15 years
21	16 years
22	17 years
23	18 years
24	19 years
25	15 to 19 years
26	20 years
27	21 years
28	22 years
29	23 years
30	24 years
31	20 to 24 years
32	25 years
33	26 years
34	27 years
35	28 years
36	29 years
37	25 to 29 years
38	30 years
39	31 years
40	32 years
41	33 years
42	34 years
43	30 to 34 years

44	35 years
45	36 years
46	37 years
47	38 years
48	39 years
49	35 to 39 years
50	40 years
51	41 years
52	42 years
53	43 years
54	44 years
55	40 to 44 years
56	45 years
57	46 years
58	47 years
59	48 years
60	49 years
61	45 to 49 years
62	50 years
63	51 years
64	52 years
65	53 years
66	54 years
67	50 to 54 years
68	55 years
69	56 years
70	57 years
71	58 years
72	59 years
73	55 to 59 years
74	60 years
75	61 years
76	62 years
77	63 years
78	64 years
79	60 to 64 years
80	65 years
81	66 years
82	67 years
83	68 years
84	69 years
85	65 to 69 years
86	70 to 74 years

87	75 to 79 years
88	80 to 84 years
89	85 to 89 years
90	90 years and over
91	0 to 14 years
92	0 to 15 years
93	0 to 16 years
94	0 to 17 years
95	15 to 49 years
96	15 to 64 years
97	16 to 64 years
98	17 to 64 years
99	18 to 24 years
100	18 to 64 years
101	18 years and over
102	25 to 44 years
103	45 to 64 years
104	65 years and over
105	Median age
106	70 years
107	71 years
108	72 years
109	73 years
110	74 years
111	75 years
112	76 years
113	77 years
114	78 years
115	79 years
116	80 years
117	81 years
118	82 years
119	83 years
120	84 years
121	85 years
122	86 years
123	87 years
124	88 years
125	89 years
126	90 to 94 years
127	90 years
128	91 years
129	92 years

130	93 years
131	94 years
132	95 to 99 years
133	95 years
134	96 years
135	97 years
136	98 years
137	99 years
138	100 years and over

Appendix 2: CURL Examples

SDMX structure sample URL:

```
curl -X GET -k -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/structure/Data_Structure_13100101'
```

```
curl -X GET -k -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/structure/Data_Structure_13100101'
```

pid=13100101

```
curl -X GET -k -H 'Accept: application/vnd.sdmx.structure+xml;version=2.1' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/structure/Data_Structure_13100101'
```

```
curl -X GET -k -H 'Accept: application/vnd.sdmx.structure+xml;version=2.1' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/structure/Data_Structure_13100101'
```

pid=13100101

Header:

Accept: application/vnd.sdmx.structure+xml;version=2.1

```
curl -X GET -k -H 'Accept: application/xml' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/structure/Data_Structure_13100101'
```

```
curl -X GET -k -H 'Accept: application/xml' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/structure/Data_Structure_13100101'
```

pid=13100101

Header:

Accept: application/xml

SDMX data sample URL:

```
curl -X GET -k -H 'Accept: application/vnd.sdmx.structurespecificdata+xml;version=2.1' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/data/DF_13100101/1.1.1+2+3+4?startPeriod
=2014&endPeriod=2015'
```

```
curl -X GET -k -H 'Accept: application/vnd.sdmx.structurespecificdata+xml;version=2.1' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/data/DF_13100101/1.1.1+2+3+4?startPer
iod=2014&endPeriod=2015'
```

pid = 13100101

Header:

Accept: application/vnd.sdmx.structurespecificdata+xml;version=2.1

dimensions are separated by "." and members are separated by "+". The date format should be "yyyy-mm-dd". if value of "day" is missing, the default value is "01"

```
curl -X GET -k -H 'Accept: application/vnd.sdmx.data+json;version=1.0.0-wd' -i 'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/data/DF_18100002/1.1+2?startPeriod=2018-01&endPeriod=2018-05'
```

```
curl -X GET -k -H 'Accept: application/vnd.sdmx.data+json;version=1.0.0-wd' -i 'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/data/DF_18100002/1.1+2?startPeriod=2018-01&endPeriod=2018-05'
```

pid = 18100102

Header:

Accept: application/vnd.sdmx.data+json;version=1.0.0-wd

```
curl -X GET -k -H 'Accept: application/vnd.sdmx.genericdata+xml;version=2.1' -i 'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/data/DF_13100101/1.1.1+2+3+4?startPeriod=2014&endPeriod=2015'
```

```
curl -X GET -k -H 'Accept: application/vnd.sdmx.genericdata+xml;version=2.1' -i 'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/data/DF_13100101/1.1.1+2+3+4?startPeriod=2014&endPeriod=2015'
```

pid = 13100101

Header:

Accept: application/vnd.sdmx.genericdata+xml;version=2.1

```
curl -X GET -k -H 'Accept: application/xml' -i 'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/data/DF_13100101/1.1.1+2+3+4?startPeriod=2014&endPeriod=2015'
```

```
curl -X GET -k -H 'Accept: application/xml' -i 'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/data/DF_13100101/1.1.1+2+3+4?startPeriod=2014&endPeriod=2015'
```

pid = 13100101

Header:

Accept: application/xml

```
curl -X GET -k -H 'Accept: application/json' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/data/DF_13100101/1.1.1+2+3+4?startPeriod
=2014&endPeriod=2015'
```

```
curl -X GET -k -H 'Accept: application/json' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/data/DF_13100101/1.1.1+2+3+4?startPer
iod=2014&endPeriod=2015'
```

pid = 13100101

Header:

Accept: application/xml

```
curl -X GET -k -H 'Accept: application/json' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/data/DF_13100101/1.1.1+2+3+4?firstNObser
vations=1'
```

```
curl -X GET -k -H 'Accept: application/json' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/data/DF_13100101/1.1.1+2+3+4?firstNOb
servations=1'
```

```
curl -X GET -k -H 'Accept: application/xml' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/data/DF_13100101/1.1.1+2+3+4?lastNObser
vations=1'
```

```
curl -X GET -k -H 'Accept: application/xml' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/data/DF_13100101/1.1.1+2+3+4?lastNOb
servations=1'
```

Vector examples

```
curl -X GET -k -H 'Accept: application/xml' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/vector/v114809245?lastNObservations=1'
```

```
curl -X GET -k -H 'Accept: application/xml' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/vector/v114809245?lastNObservations=1'
```

```
curl -X GET -k -H 'Accept: application/json' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/rest/vector/v114809245?firstNObservations=1'
```

```
curl -X GET -k -H 'Accept: application/json' -i
'https://www150.statcan.gc.ca/t1/wds/sdmx/statcan/v1/rest/vector/v114809245?firstNObservations=1
'
```