



ECONOMIC INDICATOR TIME SERIES API USER GUIDE

*How to construct API calls and use Python to request data from the
Census Bureau Economic Indicator Time Series (EITS)*

Table of Contents

1	Introduction	5
2	Purpose & Applicable use cases.....	6
3	Access to the API.....	7
4	Constructing a data call.....	8
	Step 1: Choose a Dataset	8
	Step 2: Gather required parameters.....	9
	Step 3: Format your API call.....	10
5.	Parameters.....	12
	<i>Example #1 – Get data using category code and data type code</i>	12
	<i>Example #2 - Get data with state level data</i>	14
	<i>Example #3 - Get data for the year 2017.....</i>	16
	<i>Example #4 – Get data for the year 2017 and 3rd quarter.....</i>	17
	<i>Example #5 – Get data for the year 2000 and month of September</i>	17
	<i>Example #6 – Get data since 2005.....</i>	18
	<i>Example #7 – Get data before 2002 second quarter.....</i>	18
	<i>Example #8 – Get data between those two dates</i>	19
6.	Common Errors	20
	<i>Example error #1 – No & before a variable with a value</i>	20
	<i>Example error #2 – Missing ‘time’ value</i>	20
	<i>Example error #3 – ‘time’ value outside of range</i>	21
	<i>Example error #4 – Invalid quarter value for ‘time’</i>	21
	<i>Example error #5 – Wrong ‘month’ value</i>	22
	<i>Example error #6 – Missing the ‘for’ argument</i>	22
7.	How to use the API with Python	24
	Step 1: Open your IDE or Text Editor	24
	Step 2: Add libraries.....	24
	Step 3: Set up your API key	25
	Step 4: Setup base URL	25
	Step 5: Create placeholders for parameters.....	25
	Step 6: Create a dictionary list of parameters	25
	Step 7: Add parameters to URL.....	26
	Step 8: Pull and read API data.....	26

Economic Indicators Time Series API User Guide

Step 9: Transfer the data to JSON format..... 26

Step 10: Print the JSON formatted data 27

8. Questions 28

9. Additional resources 28

Census Data API Developers Page 28

Convert JSON to CSV data..... 28

10. References 28

Python for Everybody 28

Census Data API User Guide 28

Appendix A – Indicator information..... 29

Appendix B – Python example code 31

List of Tables

Table 1: Indicator Dataset	8
Table 2: Parameters	10

List of Figures

Figure 1: Request a key	7
Figure 2: Activate key email	7
Figure 3: Construction Data Call URL	8
Figure 4: API Call Format	11

1 INTRODUCTION

The Census Bureau conducts hundreds of business surveys each year to provide official statistics for the U.S. economy. The statistics provide critical data that drives policymakers, investors, and the American public to make crucial decisions daily. Economic indicators are monthly and quarterly surveys that provide the timeliest data available and show trends across the core industries that comprise the U.S. economy. These surveys set the standard for U.S. economic statistics and are fueled by the data provided by individual businesses. The data offers reliable comprehensive measures of the U.S. economy. A variety of statistics are derived from the surveys covering construction, housing, international trade, retail trade, wholesale trade, services, and manufacturing. The data collected in each survey provides measures of economic activity that allows the analysis of economic performance and informs business investment and policy decisions. Visit the Census Bureau's [Business and Economy website](#) to learn more information.

2 PURPOSE & APPLICABLE USE CASES

In addition to the press releases and data tables that are provided on the [Economic Indicators Briefing Room](#) of the Census website, the Census Bureau publishes data sets in its Application Programming Interface (API). The API presents the data in a standard format and allows for quick access to specific data sets. Benefits to using the API include:

- Creation of queries that will automatically update as new data is released.
- Machine readable format for ingestion into third-party applications.
- Ability to combine and analyze multiple similar datasets.

This guide will familiarize you with the data available in the Census Bureau Economic Indicator Time Series (EITS), how to construct API calls, and how to request data using Python. This guide assumes the following:

- Familiarity with the concepts and techniques of retrieving data from Web Services.
- Familiarity with the JavaScript Notation (JSON) and comma separated value (CSV) data.
- Familiarity with IDE's (Integrated Development Environment) or text editors for the Python section of this guide.

Users of the Economic Time Series API can retrieve a wide range of data.

- For example, if you needed information from the Housing Vacancies and Homeownership Survey, an API call could retrieve the most current survey data for this year, any given year, or even a specific quarter of a year, provided that data has been collected and made available to the public. In this example, Housing Vacancies and Homeownership data starts in 1956, so you have decades of data sets available. Any calls prior to 1956 would not produce any data.
- In a similar case, if you needed the information from the New Residential Construction survey, you could retrieve data for any year or specific month in a year, dating back to 1959.

While Census Bureau provides raw data, you can combine multiple datasets to get an overarching perspective of the economy, and the correlations driving the statistics to allow more in-depth analysis.

- For instance, if you wanted to make a prediction about whether Monthly Wholesale Trade Sales correlate to Monthly Retail Trade, you could analyze both datasets and look for trends. Using a data analytics or visualization program, you can contrast whether wholesale trade sales are positively or negatively affected when retail sales are high. The data is returned in JSON format which allows it to be parsed by several programming languages to format the output in the manner you desire.

3 ACCESS TO THE API

The Census Bureau's [Developers website](#) is the portal to the API. The API output is in JavaScript Object Notation (JSON) format. The API allows you to make 500 calls daily without the need for an API key. If you need more than 500 daily calls, you will need an API key which can be obtained via the [API Key request](#). When selecting the API Key request link, you should see this form:



Request A Key

Organization Name:

Email Address:

☐ I agree to the [terms of service](#)

Figure 1: Request a key

Once completed, you will receive an email with an activation link within minutes of your submission (see screenshot example of confirmation email below). The link in the email must be clicked to activate your key. If the activation link email is not received within an hour, please email cedsci.feedback@census.gov.



Figure 2: Activate key email

4 CONSTRUCTING A DATA CALL

All Time Series data start with a base URL:

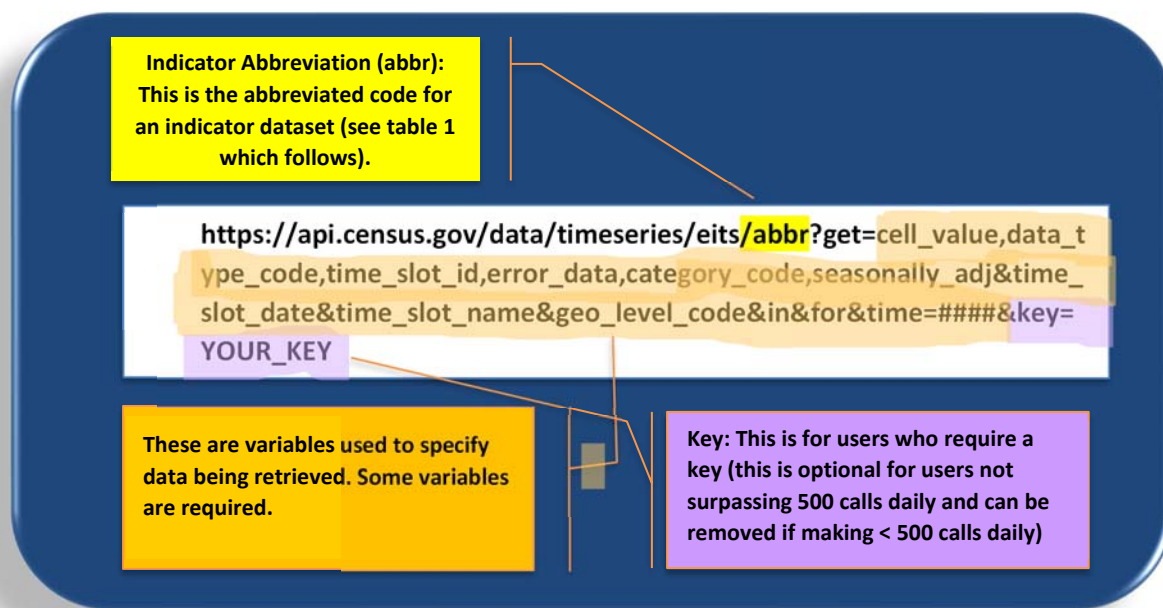


Figure 3: Construction a Data Call

STEP 1: CHOOSE A DATASET

To make an API call, first start with this base: <https://api.census.gov/data/timeseries/eits/>

Then, choose the indicator dataset that you want to retrieve data from. Use the chart below to get the abbreviated code for your respective indicator.

Table 1: Indicator Dataset

Indicator Dataset	Abbreviation
Construction Spending (VIP)	vip
New Residential Sales (NRS)	ressales
New Residential Construction (NRC)	resconst
Monthly Wholesale Trade Inventories (MWTS)	mwtis
Advance Monthly Retail Sales (MARTS)	marts
Monthly Trade & Inventory Sales (MTIS)	mtis
Monthly Retail Sales (MRTS)	mrts
Advance Report on Durable Goods (ADVM3)	advdm3
Manufacturer's Shipments Inventories & Orders (M3)	m3
International Trade (FTD)	ftd
Housing Vacancies & Homeownership Rate (HV)	hv

Quarterly Services (QSS)	qss
Quarterly Financial Report (QFR)	qfr
Business Formation Statistics (BFS)	bfs
Non-Indicators Dataset	
Manufactured Homes (MHS)(1959 – 2014)	mhs
Manufactured Housing (MHS2) (2014-Present)	mhs2
Quarterly Survey of Public Pensions (QPR)	qpr
Quarterly Summary of State and Local Taxes (QTAX)	qtax

For example, to make a call on the International Trade Indicator dataset, your code so far would look like this:

<https://api.census.gov/data/timeseries/eits/ftd>

The code is not yet complete; next you will need add parameters.

STEP 2: GATHER REQUIRED PARAMETERS

Once you have your starting code and indicator dataset abbreviation, you will need to define parameters. The first parameter is **time_slot_id** followed by **cell_value**, **error_data**, **seasonally_adj**, **category_code**, **data_type_code**, **for** and **time**.

The parameter section of the code looks like this (highlighted yellow area):

https://api.census.gov/data/timeseries/eits/ftd?get=cell_value,time_slot_id,error_data,category_code&seasonally_adj&data_type_code&for=US&time=YEAR

Continuing our example of a data call for the International Trade Indicator, if you wanted to pull data just from the year 1992, your code would look like this:

https://api.census.gov/data/timeseries/eits/ftd?get=cell_value,time_slot_id,error_data,category_code&seasonally_adj&data_type_code&for=US&time=1992

Defining parameters in the API call allows for direct access to the data without having to scroll through lengthy data tables. Parameters are like filters. The parameter variables will specify what data should be returned in the API call. Some parameters are required, some are not. Some parameter values are required, some are not. When a parameter is **required**, it must be included in the query string with or without a value. The most common required parameters are **‘for’** and **‘time.’** A parameter with a **required value** must be included in the query string with a value. The most common parameter values for **‘for’** are US (to pull data for all US states); for **‘time’** the most common parameters values would be a year, quarter in a specific year, or month in a specific year. The more parameters with values added to them in the query string, the more the data filters for your selected values.

Adding a value to a parameter filters the indicator time series data. Only include values when you are looking for specific sets of data, eliminating unwanted data. There are 13 variables for all indicator time series data.

Table 2: Parameters

Name	Label	Concept	Required	Predicate Type
category_code	Industry list	Required Variables	Required	string
cell_value	Data value	Required Variables	Required	string
data_type_code	Item type	Required Variables	Required	string
error_data	Error data yes or no	Required Variables	Not Required	string
for	Census API FIPS 'for' clause	Census API Geography Specification	Predicate-only	fips-for
geo_level_code	Geo level code	Required Variables	Not Required	string
in	Census API FIPS 'in' clause	Census API Geography Specification	Predicate-only	fips-in
program_code	Component Name	Required Variables	Default displayed	string
seasonally_adj	Seasonally adjusted yes or no	Required Variables	Required	string
time	ISO-8601 Date/Time Value	Census API Date/Time Specification	Required, Predicate-only	datetime
time_slot_date	Time Slot Data	Required Variables	Not Required	string
time_slot_id	Time Slot	Required Variables	Required	string
time_slot_name	Time Slot Name	Required Variables	Not Required	string
13 Variables				

Which parameters and parameter values are required? [Appendix A](#) provides URLs for all the variables in each parameter that will be needed to build the API call.

STEP 3: FORMAT YOUR API CALL

Take the base URL for your indicator time series, then add the respective parameters (making sure to include a value for the parameters that require it). When adding parameters, the order that you place them in the query string reflects how the data is returned. For instance, if you were trying to construct a call for the BFS time series pulling only data from 2012, the URL would look like this.

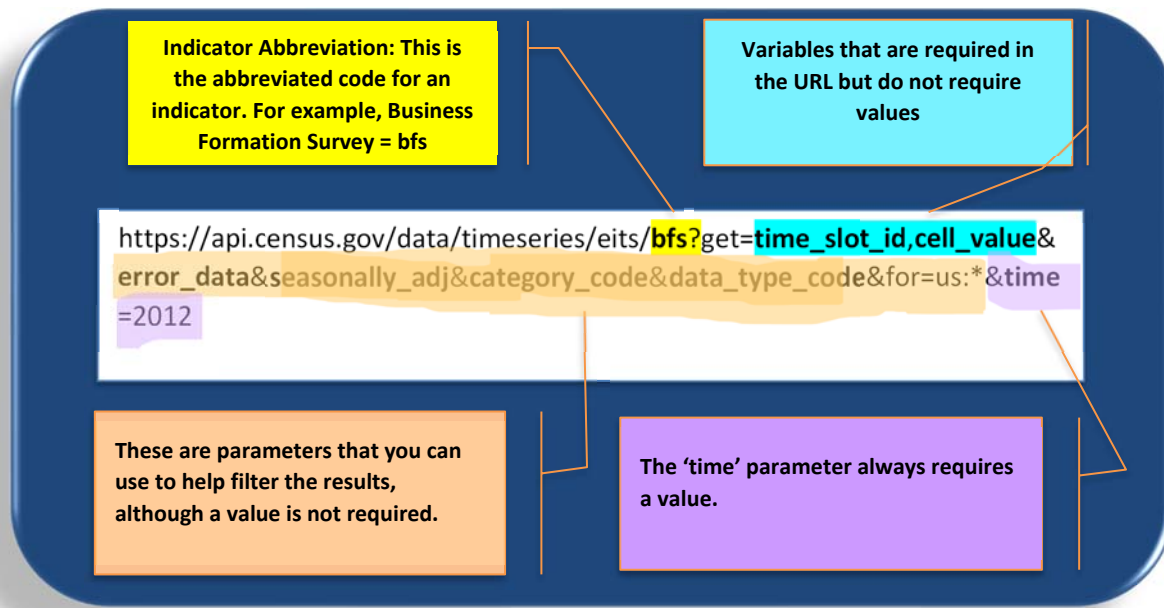


Figure 4: API Call Format

As previously noted, the first parameter is **time_slot_id** followed by **cell_value**, **error_data**, **seasonally_adj**, **category_code**, **data_type_code**, **for** and **time**. The output in JavaScript Object Notation (JSON) format for the first row of column labels would print out in the same order and the corresponding data would fall under those columns. This feature allows you to specify the format in which the data is returned. Only one call can be made at once.

Once a call is made, the API will return the following result:

```
[["data_type_code","time_slot_id","seasonally_adj","category_code","cell_value","error_data","time","us"],
["T","1329","no","BA_BA","691827","no","2012-01","1"],
["T","1329","yes","BA_BA","618671","no","2012-01","1"],
["T","1329","no","BA_CBA","143420","no","2012-01","1"],
["T","1329","yes","BA_CBA","130895","no","2012-01","1"],
["T","1329","no","BA_HBA","320848","no","2012-01","1"],
["T","1329","yes","BA_HBA","286947","no","2012-01","1"],
["T","1329","no","BA_WBA","138487","no","2012-01","1"],
```

5. PARAMETERS

All variables that have a **‘required’** attribute must be included in the query string with or without a value. All datasets require the **‘time’** parameter to have a value.

- The **‘time’** value can be a 4-digit year for all indicators in the format **####**.
- For quarterly surveys, the format can be filtered further by adding **–Q1**, **–Q2**, **–Q3**, or **–Q4** to the four-digit year value. For example:
 - **2008** would pull the data for the entire year, but . . .
 - **2008-Q1** would pull data for the first quarter of 2008
 - **2008-Q2** would pull data for the second quarter of 2008, etc.
- For monthly surveys, the format can be filtered further using **####-\$\$**, where **\$\$** represents the 2-digit month. For example:
 - **2010-11** would pull data only from November of 2010
 - **2012-01** would pull data only from January of 2012
 - **2012** again would only pull data for the entire year of 2012
- Using **‘from’** before a number, formatted **‘from+####’**, will allow you to retrieve data starting from January of a particular year to present.
- Using **‘to’** before a number, formatted **‘to+####-##’**, will allow you to pull data before and up until that date. For example, **to+2007-04** would pull all survey data up to April 2007.
- Using **‘from’** and **‘to’** together, with dates for both, formatted **‘from+####-##+to+####-##’** will produce data between those dates. Noting that **‘+’** is the URL escape character for a space.
- Using **‘*’** for the value will pull all data for that parameter. This is useful for small datasets but will throw an error if you attempt to pull all the data from a dataset containing large amounts of data. The server will not allow one query to tax all the available resources.

Parameters **‘error_data’** and **‘seasonally_adj’** have values of **‘yes’** or **‘no’**.

Parameters **‘cell_value’**, **‘time_slot_name’** and **‘time_slot_date’** do not require a value for any dataset.

Parameters **‘category_code’** and **‘data_type_code’** have numerous values that can be used to help refine the data call. Links to all possible values for each parameter are included in the respective [Appendix A](#) for each dataset. Getting each datasets’ api xml data will list all the possible values for **‘category_code’**, **‘unit_code’**, **‘geo_level_code’** and **‘data_type_code’**.

EXAMPLE #1 – GET DATA USING CATEGORY CODE AND DATA TYPE CODE

Here is a look at the xml file for Advance Monthly Sales for Retail and Food Services ([MARTS](#)). Each variable has all the listed values and their description. The **‘query_params’** section will help you see the relationship between the **‘category_code’** and **‘data_type_code’**, as it lists all possible queries for the respective dataset. The **‘unit_code’** describes what units will be represented in the returned **‘cell_value’** variable. The **‘from’** parameter shows the start of the dataset. The **‘to’** parameter shows the end of the dataset.

Here are the steps for gathering your parameter values. In this example, you are interested in getting the data about ‘Monthly Sales of Motor Vehicle and Parts Dealers’ in the year 2000:

1. You would look at the **MARTS** xml file (found in [Appendix A](#)) to find the code for ‘Motor Vehicle and Parts Dealers’, which is ‘441’, and use that value for ‘**category_code**’.
2. Then in the same xml file you would find the code for ‘Sales – Monthly’, which is ‘SM’, and use that value for ‘**data_type_code**’.
3. Next recall that you want data from 2000, you use this value for ‘**time**’.

The arrows in the screenshots point to the codes used for the example.

```

<program code="MARTS" description="Advance Monthly Sales for Retail and Food Services" release_freq="monthly">
  <categories>
    <category code="44X72" description="44X72: Retail Trade and Food Services"/>
    <category code="44Y72" description="44Y72: Retail Trade and Food Services, ex Auto"/>
    <category code="44Z72" description="44Z72: Retail Trade and Food Services, ex Gas"/>
    <category code="44W72" description="44W72: Retail Trade and Food Services, ex Auto and Gas"/>
    <category code="44000" description="44000: Retail Trade"/>
    <category code="441" description="441: Motor Vehicle and Parts Dealers"/>
    <category code="441X" description="441X,4412: Auto and Other Motor Vehicles"/>
    <category code="442" description="442: Furniture and Home Furnishings Stores"/>
  </categories>
  <data_types>
    <data_type code="SM" description="Sales - Monthly" unit_code="MLNS"/>
    <data_type code="MPCSM" description="Sales - Monthly Percent Change" unit_code="PCT"/>
    <data_type code="E_SM" description="CV of Sales - Monthly" unit_code="PCT"/>
    <data_type code="E_MPCSM" description="SE of Sales - Monthly Percent Change" unit_code="PCT"/>
  </data_types>
  <query_params>
    <query_param category_code="44X72" data_type_code="SM" geo_level_code="US" error_data="no" from="1992" to="2021" seasonally_adj="no/yes"/>
    <query_param category_code="44X72" data_type_code="E_SM" geo_level_code="US" error_data="yes" from="2001" to="2021" seasonally_adj="yes/no"/>
    <query_param category_code="44X72" data_type_code="MPCSM" geo_level_code="US" error_data="no" from="1992" to="2021" seasonally_adj="yes/no"/>
    <query_param category_code="44X72" data_type_code="E_MPCSM" geo_level_code="US" error_data="yes" from="2001" to="2021" seasonally_adj="yes/no"/>
    <query_param category_code="44Y72" data_type_code="SM" geo_level_code="US" error_data="no" from="1992" to="2021" seasonally_adj="no/yes"/>
    <query_param category_code="44Y72" data_type_code="E_SM" geo_level_code="US" error_data="yes" from="2001" to="2021" seasonally_adj="yes/no"/>
    <query_param category_code="44Y72" data_type_code="MPCSM" geo_level_code="US" error_data="no" from="1992" to="2021" seasonally_adj="yes/no"/>
    <query_param category_code="44Y72" data_type_code="E_MPCSM" geo_level_code="US" error_data="yes" from="2001" to="2021" seasonally_adj="yes/no"/>
    <query_param category_code="44000" data_type_code="SM" geo_level_code="US" error_data="no" from="1992" to="2021" seasonally_adj="no/yes"/>
    <query_param category_code="44000" data_type_code="E_SM" geo_level_code="US" error_data="yes" from="2001" to="2021" seasonally_adj="yes/no"/>
    <query_param category_code="44000" data_type_code="MPCSM" geo_level_code="US" error_data="no" from="1992" to="2021" seasonally_adj="yes/no"/>
    <query_param category_code="44000" data_type_code="E_MPCSM" geo_level_code="US" error_data="yes" from="2001" to="2021" seasonally_adj="yes/no"/>
    <query_param category_code="441" data_type_code="SM" geo_level_code="US" error_data="no" from="1992" to="2021" seasonally_adj="no/yes"/>
    <query_param category_code="441" data_type_code="E_SM" geo_level_code="US" error_data="yes" from="2001" to="2021" seasonally_adj="yes/no"/>
    <query_param category_code="441" data_type_code="MPCSM" geo_level_code="US" error_data="no" from="1992" to="2021" seasonally_adj="yes/no"/>
  </query_params>
</program>

```

After finding the codes, you now can include the values in the query string. Here is the resulting query string:

```
https://api.census.gov/data/timeseries/eits/marts?get=cell_value,time_slot_id,error_data&category_code=441
&seasonally_adj&data_type_code=SM&time=2000
```

Produces this output:

Economic Indicators Time Series API User Guide

```
[["cell_value","time_slot_id","error_data","category_code","seasonally_adj","data_type_code","time"],
["58773","601","no","441","no","SM","2000-01"],
["67679","601","no","441","yes","SM","2000-01"],
["66892","602","no","441","no","SM","2000-02"],
["69085","602","no","441","yes","SM","2000-02"],
["75067","603","no","441","no","SM","2000-03"],
["68433","603","no","441","yes","SM","2000-03"],
["65986","604","no","441","no","SM","2000-04"],
["65812","604","no","441","yes","SM","2000-04"],
["73050","605","no","441","no","SM","2000-05"],
["65781","605","no","441","yes","SM","2000-05"],
["73002","606","no","441","no","SM","2000-06"],
["66638","606","no","441","yes","SM","2000-06"],
["67260","607","no","441","no","SM","2000-07"],
["65205","607","no","441","yes","SM","2000-07"],
["71850","608","no","441","no","SM","2000-08"],
["65311","608","no","441","yes","SM","2000-08"],
["65835","609","no","441","no","SM","2000-09"],
["66850","609","no","441","yes","SM","2000-09"],
["63805","610","no","441","no","SM","2000-10"],
["66208","610","no","441","yes","SM","2000-10"],
["58449","611","no","441","no","SM","2000-11"],
["64645","611","no","441","yes","SM","2000-11"],
["56241","612","no","441","no","SM","2000-12"],
["63524","612","no","441","yes","SM","2000-12"]]
```

EXAMPLE #2 - GET DATA WITH STATE LEVEL DATA

Here is an example using the Quarterly Summary of State and Local Taxes (QTAX) time series data. QTAX and Business Formation Statistics (BFS) datasets have numerous values for the '**geo_level_code**', those relationships can also be found in their respective xml files. QTAX and BFS are the only indicators that provide state level data. Note that although they provide many values for '**geo_level_code**', the only 2 values accepted are 'US', for national data, and '*' for all data. To retrieve data about the Latest State Tax Collections by State for Tobacco Products Sales, look at the xml document to get the parameter values associated with those descriptions. The arrows in the screenshots point to the codes used for the example.

```

<program code="QTAX" description="Quarterly Summary of State & Local Taxes" release_freq="quarterly">
  <categories>
    <category code="QTAXCAT1" description="Table 1 - Latest National Totals of State and Local Tax Revenue"/>
    <category code="QTAXCAT2" description="Table 2 - Latest National Totals of State Tax Revenue"/>
    <category code="QTAXCAT3" description="Table 3 - Latest State Tax Collections by State and Type of Tax"/>
  </categories>
  <units>
    <unit code="MLNS" description="Millions of Dollars"/>
    <unit code="BLNS" description="Billions of Dollars"/>
    <unit code="KS" description="Thousands of Dollars"/>
    <unit code="PCT" description="Percent"/>
    <unit code="UNITS" description="Units"/>
    <unit code="RATIO" description="Ratio"/>
    <unit code="MO" description="Number of Months"/>
    <unit code="DOL" description="Dollars"/>
    <unit code="K" description="Thousands of Units"/>
    <unit code="CPS" description="Cents per dollar"/>
    <unit code="%PTS" description="Percentage Points"/>
    <unit code="CENTS" description="Cents"/>
    <unit code="PCTCHG" description="% change"/>
    <unit code="PCTCHGINV" description="% change in Inventories"/>
    <unit code="QT" description="Number of Quarters"/>
  </units>
  <data_types>
    <data_type code="T01" description="T01 Property Taxes" unit_code="MLNS"/>
    <data_type code="T014QE" description="T01 Property Taxes, 4 Quarters Ending" unit_code="MLNS"/>
    <data_type code="T09" description="T09 General Sales and Gross Receipts Taxes" unit_code="MLNS"/>
    <data_type code="T094QE" description="T09 General Sales and Gross Receipts Taxes, 4 Quarters Ending" unit_code="MLNS"/>
    <data_type code="T10" description="T10 Alcoholic Beverages Sales Tax" unit_code="MLNS"/>
    <data_type code="T104QE" description="T10 Alcoholic Beverages Sales Tax, 4 Quarters Ending" unit_code="MLNS"/>
    <data_type code="T11" description="T11 Amusements Sales Tax" unit_code="MLNS"/>
    <data_type code="T12" description="T12 Insurance Premiums Sales Tax" unit_code="MLNS"/>
    <data_type code="T13" description="T13 Motor Fuels Sales Tax" unit_code="MLNS"/>
    <data_type code="T134QE" description="T13 Motor Fuels Sales Tax, 4 Quarters Ending" unit_code="MLNS"/>
    <data_type code="T14" description="T14 Pari-Mutuels Sales Tax" unit_code="MLNS"/>
    <data_type code="T15" description="T15 Public Utilities Sales Tax" unit_code="MLNS"/>
    <data_type code="T16" description="T16 Tobacco Products Sales Tax" unit_code="MLNS"/>
    <data_type code="T164QE" description="T16 Tobacco Products Sales Tax, 4 Quarters Ending" unit_code="MLNS"/>
    <data_type code="T19" description="T19 Other Selective Sales and Gross Receipts Taxes" unit_code="MLNS"/>
    <data_type code="T20" description="T20 Alcoholic Beverages License" unit_code="MLNS"/>
  </data_types>
</program>

```

1. The 'category_code' for Latest State Tax Collections by State and Type of Tax is 'QTAXCAT3'.
2. The 'data_category_code' for Tobacco Products Sales Tax is 'T16'.
3. The value for 'geo_level_code' is '*', which retrieves all state data from the that dataset.
4. The 'time' value for this example is 2000.

Here is the resulting query string:

```

https://api.census.gov/data/timeseries/eits/ntax?get=cell_value,time_slot_id,error_data&geo_level_code=*&category_code=QTAXCAT3&seasonally_adj&data_type_code=T16&time=2000

```

Produces this output:

Economic Indicators Time Series API User Guide

```
[["cell_value","time_slot_id","error_data","geo_level_code","category_code","seasonally_adj","data_type_code","time"],
["10","1281","no","AK","QTAXCAT3","no","T16","2000-Q1"],
["16","1281","no","AL","QTAXCAT3","no","T16","2000-Q1"],
["22","1281","no","AR","QTAXCAT3","no","T16","2000-Q1"],
["37","1281","no","AZ","QTAXCAT3","no","T16","2000-Q1"],
["120","1281","no","CA","QTAXCAT3","no","T16","2000-Q1"],
["16","1281","no","CO","QTAXCAT3","no","T16","2000-Q1"],
["28","1281","no","CT","QTAXCAT3","no","T16","2000-Q1"],
["3","1281","no","DC","QTAXCAT3","no","T16","2000-Q1"],
["6","1281","no","DE","QTAXCAT3","no","T16","2000-Q1"],
["102","1281","no","FL","QTAXCAT3","no","T16","2000-Q1"],
["11","1281","no","GA","QTAXCAT3","no","T16","2000-Q1"],
["11","1281","no","HI","QTAXCAT3","no","T16","2000-Q1"],
["23","1281","no","IA","QTAXCAT3","no","T16","2000-Q1"],
["7","1281","no","ID","QTAXCAT3","no","T16","2000-Q1"],
["107","1281","no","IL","QTAXCAT3","no","T16","2000-Q1"],
["29","1281","no","IN","QTAXCAT3","no","T16","2000-Q1"],
["12","1281","no","KS","QTAXCAT3","no","T16","2000-Q1"],
["3","1281","no","KY","QTAXCAT3","no","T16","2000-Q1"],
["21","1281","no","LA","QTAXCAT3","no","T16","2000-Q1"],
["63","1281","no","MA","QTAXCAT3","no","T16","2000-Q1"],
["48","1281","no","MD","QTAXCAT3","no","T16","2000-Q1"],
```

EXAMPLE #3 - GET DATA FOR THE YEAR 2017

Here is an example using the Quarterly Services Survey time series data, which reports quarterly. The ‘time’ value has the format of #####. This will result in the API returning all data for that entire year. Here is the resulting query string to pull all data from that survey from 2017:

```
https://api.census.gov/data/timeseries/eits/qss?get=cell\_value,time\_slot\_id,seasonally\_adj&data\_type\_code&category\_code&time=2017
```

Produces this output:

```
[["cell_value","time_slot_id","seasonally_adj","data_type_code","category_code","time"],
["-1.4","1349","no","PQREV","000000A","2017-Q1"],
["3553741","1349","no","QREV","000000A","2017-Q1"],
["1.4","1349","no","E_QREV","000000A","2017-Q1"],
["0.3","1349","no","E_PQREV","000000A","2017-Q1"],
["1.4","1349","yes","PQREV","000000A","2017-Q1"],
["3611525","1349","yes","QREV","000000A","2017-Q1"],
["1.4","1349","yes","E_QREV","000000A","2017-Q1"],
["0.3","1349","yes","E_PQREV","000000A","2017-Q1"],
["Z","1349","no","PQREV","2211T","2017-Q1"],
["110046","1349","no","QREV","2211T","2017-Q1"],
["5.0","1349","no","E_QREV","2211T","2017-Q1"],
["0.7","1349","no","E_PQREV","2211T","2017-Q1"],
["-2.1","1349","yes","PQREV","2211T","2017-Q1"],
["114631","1349","yes","QREV","2211T","2017-Q1"],
["5.0","1349","yes","E_QREV","2211T","2017-Q1"],
["0.7","1349","yes","E_PQREV","2211T","2017-Q1"],
```


EXAMPLE #4 – GET DATA FOR THE YEAR 2017 AND 3RD QUARTER

Adding a specific quarter to example #3 above will produce results from that quarter. Here is the query string:

https://api.census.gov/data/timeseries/eits/qss?get=cell_value,time_slot_id,seasonally_adj&data_type_code&category_code&time=2017-Q3

Produces this output:

```
[["cell_value","time_slot_id","seasonally_adj","data_type_code","category_code","time"],
["1.0","1351","no","PQREV","000000A","2017-Q3"],
["3682913","1351","no","QREV","000000A","2017-Q3"],
["0.1","1351","no","E_PQREV","000000A","2017-Q3"],
["0.9","1351","yes","PQREV","000000A","2017-Q3"],
["3675562","1351","yes","QREV","000000A","2017-Q3"],
["3.4","1351","no","E_QREV","2211T","2017-Q3"],
["3.4","1351","yes","E_QREV","2211T","2017-Q3"],
["13.7","1351","no","PQREV","2213T","2017-Q3"],
["0.7","1351","no","E_QREV","000000A","2017-Q3"],
["0.7","1351","yes","E_QREV","000000A","2017-Q3"],
["0.1","1351","yes","E_PQREV","000000A","2017-Q3"],
["18.9","1351","no","PQREV","2211T","2017-Q3"],
["133630","1351","no","QREV","2211T","2017-Q3"],
["-15.2","1351","no","PQREV","22121T","2017-Q3"],
["16503","1351","no","QREV","22121T","2017-Q3"],
["2.4","1351","no","E_PQREV","2211T","2017-Q3"],
["-0.9","1351","yes","PQREV","2211T","2017-Q3"],
["116606","1351","yes","QREV","2211T","2017-Q3"],
```

EXAMPLE #5 – GET DATA FOR THE YEAR 2000 AND MONTH OF SEPTEMBER

Here is an example of International Trade time series data, which reports monthly. The ‘time’ value has the format of ####-\$. This will result in the API returning data for only that month. Here is the resulting query string:

https://api.census.gov/data/timeseries/eits/ftd?get=program_code,cell_value,time_slot_id,time_slot_date,time_slot_name&data_type_code&seasonally_adj&category_code&time=2000-09

Produces this output:

```
[["program_code","cell_value","time_slot_id","time_slot_date","time_slot_name","data_type_code","seasonally_adj","category_code","time"],
["FTD","107598","609","2000-09-01 00:00:00.0","Sep2000","IMP","no","BOPG","2000-09"],
["FTD","-39861","609","2000-09-01 00:00:00.0","Sep2000","BAL","no","BOPG","2000-09"],
["FTD","67737","609","2000-09-01 00:00:00.0","Sep2000","EXP","no","BOPG","2000-09"],
["FTD","-33891","609","2000-09-01 00:00:00.0","Sep2000","BAL","yes","BOPGS","2000-09"],
["FTD","92911","609","2000-09-01 00:00:00.0","Sep2000","EXP","yes","BOPGS","2000-09"],
["FTD","126802","609","2000-09-01 00:00:00.0","Sep2000","IMP","yes","BOPGS","2000-09"]]
```

EXAMPLE #6 – GET DATA SINCE 2005

Here is an example using the same data set retrieving all the data starting from a particular year. The ‘time’ value has the format of **from+####**. Here is the resulting query string:

```
https://api.census.gov/data/timeseries/eits/ftd?get=program_code,cell_value,time_slot_id,time_slot_date,time_slot_name&data_type_code&seasonally_adj&category_code&time=from+2005
```

Produces this output:

```
[["program_code","cell_value","time_slot_id","time_slot_date","time_slot_name","data_type_code","seasonally_adj","category_code","time"],
["FTD","-57492","661","2005-01-01 00:00:00.0","Jan2005","BAL","no","BOPG","2005-01"],
["FTD","67124","661","2005-01-01 00:00:00.0","Jan2005","EXP","no","BOPG","2005-01"],
["FTD","124616","661","2005-01-01 00:00:00.0","Jan2005","IMP","no","BOPG","2005-01"],
["FTD","-56189","661","2005-01-01 00:00:00.0","Jan2005","BAL","yes","BOPGS","2005-01"],
["FTD","-54628","662","2005-02-01 00:00:00.0","Feb2005","BAL","no","BOPG","2005-02"],
["FTD","69200","662","2005-02-01 00:00:00.0","Feb2005","EXP","no","BOPG","2005-02"],
["FTD","123828","662","2005-02-01 00:00:00.0","Feb2005","IMP","no","BOPG","2005-02"],
["FTD","-58095","662","2005-02-01 00:00:00.0","Feb2005","BAL","yes","BOPGS","2005-02"]]
```

EXAMPLE #7 – GET DATA BEFORE 2002 SECOND QUARTER

Here is an example using Quarterly Financial Report time series data set. The ‘time’ value has the format of **to+####-Q#**. This will result in the API returning data before and up until that date. Here is the resulting query string:

```
https://api.census.gov/data/timeseries/eits/qfr?get=cell_value,time_slot_id,error_data,category_code&for&seasonally_adj&data_type_code&time=to+2002-Q2
```

Produces this output:

```
[["cell_value","time_slot_id","error_data","category_code","seasonally_adj","data_type_code","time"],
["12070","1284","no","311","no","NWC","2000-Q4"],
["234020","1284","no","311","no","328","2000-Q4"],
["77137","1284","no","311","no","327","2000-Q4"],
["69748","1284","no","311","no","322","2000-Q4"],
["7389","1284","no","311","no","D326","2000-Q4"],
["156883","1284","no","311","no","TL","2000-Q4"],
["25831","1284","no","311","no","320","2000-Q4"],
```

EXAMPLE #8 – GET DATA BETWEEN THOSE TWO DATES

Here is an example using the same data set, pulling all the data between two dates. The ‘time’ value has the format **from+####-Q#**+**to+####-Q#**. Here is the resulting query string:

```
https://api.census.gov/data/timeseries/eits/qfr?get=cell\_value,time\_slot\_id,error\_data,category\_code&for&seasonally\_adj&data\_type\_code&time=from+2001-Q4+to+2002-Q2
```

Produces this output:


```
[["cell_value","time_slot_id","error_data","category_code","seasonally_adj","data_type_code","time"],
["17838","1288","no","311","no","NWC","2001-Q4"],
["229244","1288","no","311","no","328","2001-Q4"],
["78590","1288","no","311","no","327","2001-Q4"],
["58738","1288","no","311","no","322","2001-Q4"],
["19852","1288","no","311","no","D326","2001-Q4"],
["150654","1288","no","311","no","TL","2001-Q4"],
```

6. COMMON ERRORS

This section will outline some of the common mistakes made while attempting to make an API call to the Economic Indicator Time Series datasets. Most of these errors are created by malformation of the URL, no inclusion of required variable value or a value for time being outside of that indicators range.

EXAMPLE ERROR #1 – NO & BEFORE A VARIABLE WITH A VALUE


Here is an example using the Quarterly Services Survey dataset. Here is the query string with the error:

`https://api.census.gov/data/timeseries/eits/qss?get=cell_value,time_slot_id,error_data,data_type_code,seasonally_adj,category_code&for,time=2003` 

Produces the error:

```
error: error: missing required variable/predicate: time
```

Notice there is a ',' before 'time' when there should be an '&'. Here is the correct format.


`https://api.census.gov/data/timeseries/eits/qss?get=cell_value,time_slot_id,error_data,data_type_code,seasonally_adj,category_code&for&time=2003` 

Produces this output:

```
[["cell_value","time_slot_id","error_data","data_type_code","seasonally_adj","category_code","time"],
["30679","1296","no","QREV","no","5112T","2003-Q4"],
["2261","1296","no","GOV","no","5112T","2003-Q4"],
["6932","1296","no","HHD","no","5112T","2003-Q4"],
["21486","1296","no","BUS","no","5112T","2003-Q4"],
["5.1","1296","yes","E_QREV","no","5112T","2003-Q4"],
```

EXAMPLE ERROR #2 – MISSING 'TIME' VALUE

Here is an example using the same dataset without a value for 'time'. Here is the query string with the error.

`https://api.census.gov/data/timeseries/eits/qss?get=cell_value,time_slot_id,error_data,data_type_code,seasonally_adj,category_code&for&time` 

Produces the error:

```
error: error: date/time predicate may not be empty: time
```

The 'time' variable requires a value for all datasets. Here is the correct format.

`https://api.census.gov/data/timeseries/eits/qss?get=cell_value,time_slot_id,error_data,data_type_code,season`


`ally_adj,category_code&for&time=2010` 

Produces this output:

```
[["cell_value","time_slot_id","error_data","data_type_code","seasonally_adj","category_code","time"],
["103363","1321","no","QREV","no","2211T","2010-Q1"],
["7.8","1321","yes","E_QREV","no","2211T","2010-Q1"],
["42722","1321","no","QREV","no","22121T","2010-Q1"],
["-2.2","1321","no","PQREV","no","4841T","2010-Q1"],
["1.7","1321","yes","E_PQREV","no","4842T","2010-Q1"],
```

EXAMPLE ERROR #3 – ‘TIME’ VALUE OUTSIDE OF RANGE


Here is an example using the same dataset with an invalid ‘time’ value. Here is the query string with the error.

`https://api.census.gov/data/timeseries/eits/qss?get=cell_value,time_slot_id,error_data,data_type_code,seasonally_adj,category_code&for&time=2000` 

Notice the ‘time’ value is set to 2000. We know from [Appendix A](#) that the time series for Quarterly Services Survey starts in 2003, so the value must be set on or after that year date or no data will be returned.

EXAMPLE ERROR #4 – INVALID QUARTER VALUE FOR ‘TIME’


Here is an example using the same dataset with an invalid quarter value for the ‘time’ parameter. Here is the query string with the error.

`https://api.census.gov/data/timeseries/eits/qss?get=cell_value,time_slot_id,error_data,data_type_code,seasonally_adj,category_code&for&time=2004-01` 

Produces the error:

`error: unexpected month in date/time predicate: time`

Notice there is a numeric month value, where there should be a quarter value for the time variable. Providing a quarter value with ‘Q1’ – ‘Q4’ (Q must be capitalized) will produce data for that quarter. Here is the correct format:

`https://api.census.gov/data/timeseries/eits/qss?get=cell_value,time_slot_id,error_data,data_type_code,seasonally_adj,category_code&for&time=2004-Q1` 


Produces this output:

```
[["cell_value","time_slot_id","error_data","data_type_code","seasonally_adj","category_code","time"],
["26439","1297","no","QREV","no","5112T","2004-Q1"],
["-13.8","1297","no","PQREV","no","5112T","2004-Q1"],
["-15.1","1297","no","PBUS","no","5112T","2004-Q1"],
["-11.2","1297","no","PGOV","no","5112T","2004-Q1"],
["-10.8","1297","no","PHHD","no","5112T","2004-Q1"],
```

EXAMPLE ERROR #5 – WRONG ‘MONTH’ VALUE

Here is an example using International Trade time series data, which reports monthly. Here is the query string with the error.

```
https://api.census.gov/data/timeseries/eits/ftd?get=program_code,cell_value,time_slot_id,time_slot_date,time_slot_name&data_type_code&seasonally_adj&category_code&time=2000-Q1
```




Produces this error:

```
error: unexpected quarter in date/time predicate: time
```

Notice there is ‘Q1’ value in the time variable when there should be a month value. If you are unsure, please check [Appendix A](#), it details which surveys are monthly, quarterly and annual. Here is the correct format.

```
https://api.census.gov/data/timeseries/eits/ftd?get=program_code,cell_value,time_slot_id,time_slot_date,time_slot_name&data_type_code&seasonally_adj&category_code&time=2000-09
```




Produces this output:

```
[["program_code","cell_value","time_slot_id","time_slot_date","time_slot_name","data_type_code","seasonally_adj","category_code","time"],
["FTD","107598","609","2000-09-01 00:00:00.0","Sep2000","IMP","no","BOPG","2000-09"],
["FTD","-39861","609","2000-09-01 00:00:00.0","Sep2000","BAL","no","BOPG","2000-09"],
["FTD","67737","609","2000-09-01 00:00:00.0","Sep2000","EXP","no","BOPG","2000-09"],
["FTD","-33891","609","2000-09-01 00:00:00.0","Sep2000","BAL","yes","BOPGS","2000-09"],
["FTD","92911","609","2000-09-01 00:00:00.0","Sep2000","EXP","yes","BOPGS","2000-09"],
["FTD","126802","609","2000-09-01 00:00:00.0","Sep2000","IMP","yes","BOPGS","2000-09"]]
```

EXAMPLE ERROR #6 – MISSING THE ‘FOR’ ARGUMENT

Here is an example using the Business Formation Statistics dataset. Here is the query string with the error.

```
https://api.census.gov/data/timeseries/eits/bfs?get=cell_value,time_slot_id,error_data,category_code&seasonally_adj&data_type_code&for&time=2004
```



Produces this error:

```
error: missing 'for' argument
```

Notice there is no value for the ‘for’ variable in the query string. [Appendix A](#) describes that BFS has a required

value for the **'for'** variable. Here is the correct format.

https://api.census.gov/data/timeseries/eits/bfs?get=cell_value,time_slot_id,error_data,category_code&seasonally_adj&data_type_code&for=US&time=2004 

Produces this output:

```
[["cell_value","time_slot_id","error_data","category_code","seasonally_adj","data_type_code","time","us"],
["541170","1299","no","BA_BA","no","T","2004-03","1"],
["574777","1299","no","BA_BA","yes","T","2004-03","1"],
["206730","1299","no","BA_WBA","no","T","2004-03","1"],
["0.96","1299","no","BF_DUR4Q","no","Q","2004-03","1"],
["528290","1300","no","BA_BA","no","T","2004-04","1"],
```

These are the most common errors generated when executing an incorrect API data call. Paying close attention to the format of the URL, the required variable values and the range of the indicator's time series will eliminate most issues.

7. HOW TO USE THE API WITH PYTHON

Census API data can be called using any programming language that can send URL queries and retrieve the results. Python is a widely used language for handling big data. Python is a general-purpose programming language which you can use to produce both web and desktop applications. Python is perfect for making API call for the following reasons:

- It is easy to learn. Python focuses on simplicity and readability. It has a low learning curve, and most professors suggest it as a programmers introductory programming language.
- It is open source. Which means it is a free language with a community-based development platform. This also allows different developers to create packages and libraries that can be used throughout the community. There are many different Python libraries for data manipulation, visualization, statistics, and Machine learning.
- Widely supported. Python is a very popular language; this makes finding help online very easy. Users can visit [Stack Overflow](#) or other various online forums to get help.
- It is very versatile. Python can be used for all steps involved in ingesting, manipulating, analyzing, and visualizing the data. You can also output the data in numerous formats, allowing many different platforms to use data produced from Python.

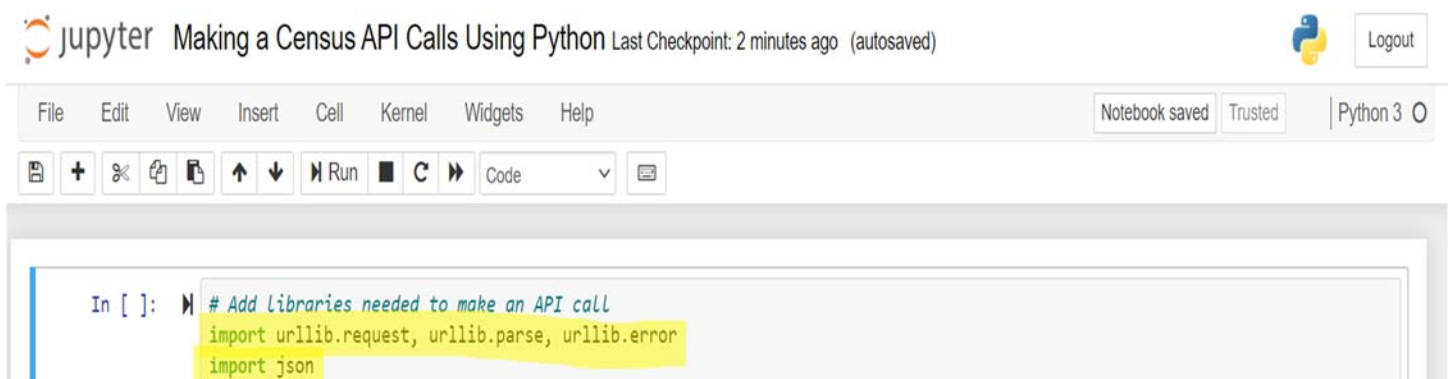
STEP 1: OPEN YOUR IDE OR TEXT EDITOR

To create a Python program, you will need a text editor or IDE (Integrated Development Environment). Some popular IDE's are Idle, PyCharm, Microsoft Visual Studio, Spyder and Eclipse. In this guide we will be using Jupyter Notebook. This example describes how to pull data from the API using the following URL:

```
https://api.census.gov/data/timeseries/eits/qfr?get=cell_value,time_slot_id&error_data=&seasonally_adj=&category_code=&data_type_code=&for=&time=from+2001-Q4+to+2002-Q2&key=4c4d5dcc24fa1b6b04a42d297883ef4934d83c0e
```

STEP 2: ADD LIBRARIES

To add the Python libraries, you will need to make the API call. This screenshot shows minimum libraries needed to perform this API call.



STEP 3: SET UP YOUR API KEY

API keys are necessary for making more than 500 calls in a 24-hour time span. The API key is set to false by default because it is not always necessary. If you have an API key, you would insert that value in single quotes to a placeholder in your program. The screenshot below shows how to setup your API key for use in the query string.

```
# By default because an API key is not needed under 500 request a day.
api_key = False
if api_key is False:
    # If you have an API key you would use it here
    api_key = '4c4d5dcc24fa1b6b04a42d297883ef4934d83c0e'
else :
    api_key = False
```

STEP 4: SETUP BASE URL

Create a placeholder for the base URL that includes the Indicator Abbreviation for the chosen data set. Include 'cell_value' and 'time_slot_id' at the end. These values never require a value, so they are attached to the base URL. This screenshot demonstrates this step.

```
# Include the base URL and the Indicator abbreviation for the chosen data set
# Include cell_value and time_slot_id at the end since will never require a value
api_url = 'https://api.census.gov/data/timeseries/eits/qfr?get=cell_value,time_slot_id'
```

STEP 5: CREATE PLACEHOLDERS FOR PARAMETERS

Create variables for the parameters required to make an API call. If you want to further filter your results, this is where you add would add values to the parameters that will further refine your query. The 'time' value is required. This screenshot illustrates this step.

```
# Create placeholder for the required parameters
# Set values for error_data, seasonally_adj, category_code, data_type_code, for, and time
error_data = '' # Optional value
seasonally_adj = '' # Optional value
category_code = '' # Optional value
data_type_code = '' # Optional value
for_param = '' # Optional value
time = 'from 2001-Q4 to 2002-Q2' # Required value
```

STEP 6: CREATE A DICTIONARY LIST OF PARAMETERS

Create a dictionary list of parameters to add to the query string. Then set them to their associated variables created in Step 5. Also, add your API key to the dictionary if needed. This screenshot demonstrates this step.

```
# Create a dictionary list of parameters to add to the query string
params = dict()
params['error_data'] = error_data
params['seasonally_adj'] = seasonally_adj
params['category_code'] = category_code
params['data_type_code'] = data_type_code
params['for'] = for_param
params['time'] = time
# Add api key to the end if you have one
if api_key is not False: params['key'] = api_key
```

STEP 7: ADD PARAMETERS TO URL

Add the dictionary list to end of your base URL using 'urlencode' to properly format the strings being added. This screenshot illustrates this step

```
# Add parameters to query string
url = api_url + urllib.parse.urlencode(params)
```

You can optionally print the URL here before pulling the data to ensure the correct URL is being used. This screenshot illustrates this optional step.

```
# Print resulting URL before getting API data
print('Retrieving', url)
```

STEP 8: PULL AND READ API DATA

Using another placeholder, submit the URL request and store it. Then using another placeholder variable, decode the data requested. This screenshot illustrates this step.

```
# Pull API Data
api_data = urllib.request.urlopen(url)
# Read API Data
data = api_data.read().decode()
print('Retrieved', len(data), 'characters')
```

*Note the print function at the end is optional code that displays how many characters were retrieved.

STEP 9: TRANSFER THE DATA TO JSON FORMAT

Now that you have retrieved the data, you must now convert the raw format into JSON format. The code in the screenshot below describes this step.

```
# Transfer data into JSON format
try:
    js = json.loads(data)
except:
    js = None

# If no JSON is found print results
if not js:
    print('==== Failure To Retrieve ====')
    print(data)

def only_dict(data):

    #Convert json string representation of dictionary to a python dict
    return ast.literal_eval(data)
```

STEP 10: PRINT THE JSON FORMATTED DATA

Now that you have retrieved the data and formatted into JSON, you can now print the results from your query. Here is the screenshot of the print function, along with the resulting JSON formatted data returned from this example.

```
print(data)
```

Produces this output:

```
Retrieving https://api.census.gov/data/timeseries/eits/qfr?get=cell_value,time_slot_id&error_data=&seasonally_adj=&category_code=&data_type_code=&for=&time=from+2001-Q4+to+2002-Q2&key=4c4d5dcc24fa1b6b04a42d297883ef4934d83c0e
Retrieved 283089 characters
[["cell_value","time_slot_id","error_data","seasonally_adj","category_code","data_type_code","time"],
["17838","1288","no","no","311","NWC","2001-Q4"],
["229244","1288","no","no","311","328","2001-Q4"],
["78590","1288","no","no","311","327","2001-Q4"],
["58738","1288","no","no","311","322","2001-Q4"],
["19852","1288","no","no","311","D326","2001-Q4"],
["150654","1288","no","no","311","TL","2001-Q4"],
["25072","1288","no","no","311","320","2001-Q4"],
["41715","1288","no","no","311","D319","2001-Q4"],
["24979","1288","no","no","311","316","2001-Q4"],
["58888","1288","no","no","311","TCL","2001-Q4"],
["17125","1288","no","no","311","D315","2001-Q4"],
["2732","1288","no","no","311","D313","2001-Q4"],
["3035","1288","no","no","311","310","2001-Q4"],
["1021","1288","no","no","311","D309","2001-Q4"],
["17959","1288","no","no","311","306","2001-Q4"],
["8949","1288","no","no","311","D304","2001-Q4"]]
```

A preview of the entire code is included in [Appendix B](#).

8. QUESTIONS

If you have any additional questions, please email us at cedsci.feedback@census.gov. Someone should respond within five business days.

9. ADDITIONAL RESOURCES

CENSUS DATA API DEVELOPERS PAGE:

<http://www.census.gov/developers/>

CONVERT JSON TO CSV DATA

<http://www.convertcsv.com/json-to-csv.htm>

10. REFERENCES

PYTHON FOR EVERYBODY

<https://www.py4e.com/>

<https://www.py4e.com/code3/geojson.py>

CENSUS DATA API USER GUIDE

<https://www.census.gov/content/dam/Census/data/developers/api-user-guide/api-guide.pdf>

APPENDIX A – INDICATOR INFORMATION

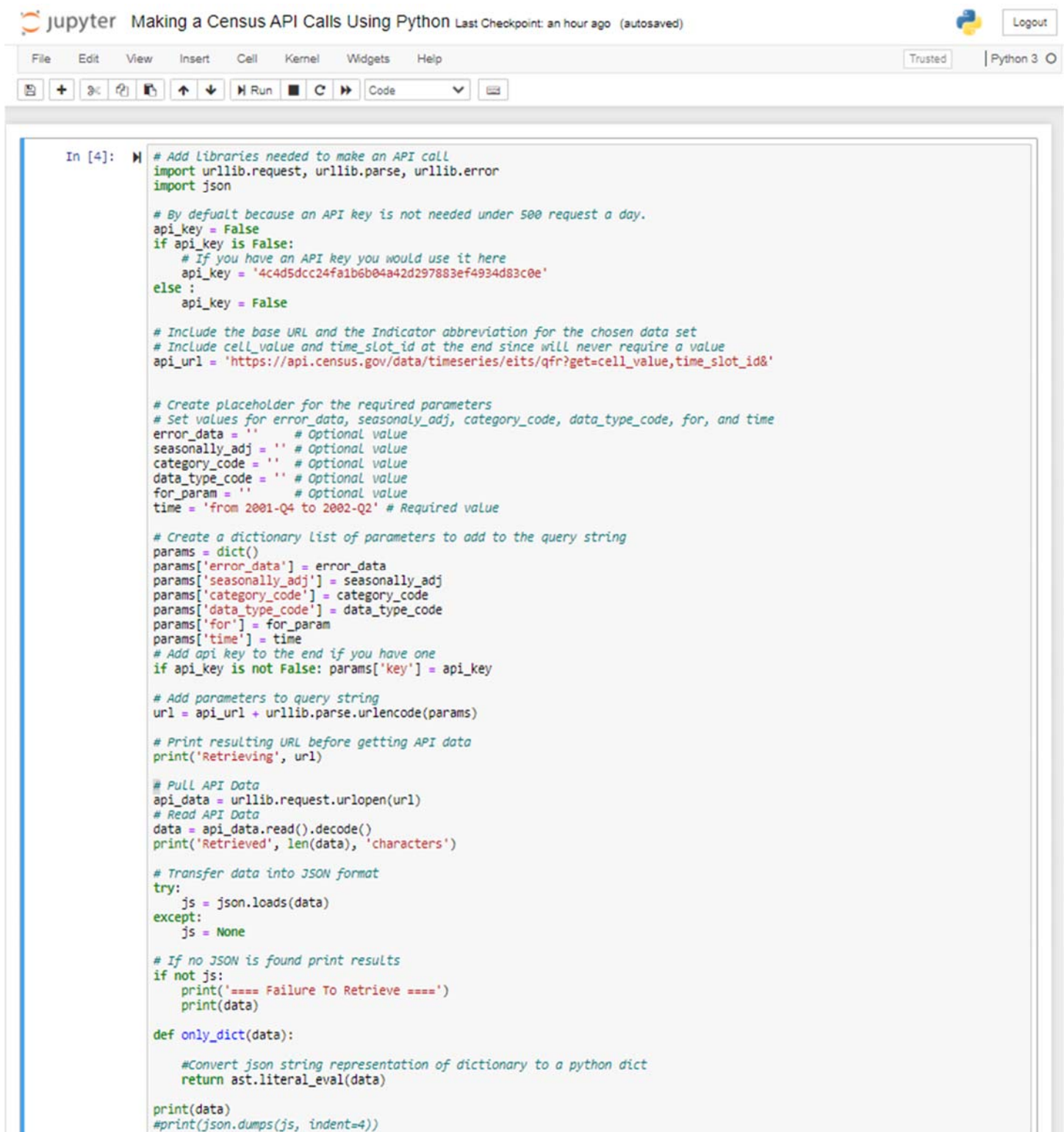
Indicator	Example URL	Parameter Variable Values	Required Parameter Value	Time Series Begins	Useful Notes
Advanced Report on Durable Goods	https://api.census.gov/data/timeseries/eits/advm3?get=cell_value,time_slot_id,error_data,category_code&seasonally_adj&data_type_code&for=US&time=1992&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/m3adv-api.xml	for, time	1992	Reports monthly.
Business Formation Statistics	https://api.census.gov/data/timeseries/eits/bfs?get=cell_value,time_slot_id,error_data,category_code&seasonally_adj&data_type_code&for=US&time=2004&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/bfs-api.xml	for, time	2004	Reports quarterly. Provides state level data. You cannot specify a state, it simply returns the data for all states.
International Trade	https://api.census.gov/data/timeseries/eits/ftd?get=cell_value,time_slot_id,error_data,category_code&seasonally_adj&data_type_code&time=1992&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/ftd-api.xml	time	1992	Reports monthly.
Housing Vacancies and Homeownership	https://api.census.gov/data/timeseries/eits/hv?get=cell_value,time_slot_id,error_data,category_code&seasonally_adj&data_type_code&time=1956&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/hv-api.xml	time	1956	Reports quarterly.
Manufacturers' Shipments, Inventories, and Orders	https://api.census.gov/data/timeseries/eits/m3?get=data_type_code,time_slot_id,seasonally_adj,category_code,cell_value,error_data&for=US&time=1992&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/m3-api.xml	for, time	1992	Reports monthly.
Advanced Monthly Sales for Retail and Food Services	https://api.census.gov/data/timeseries/eits/marts?get=cell_value,time_slot_id,error_data,category_code&seasonally_adj&data_type_code&time=1992&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/marts-api.xml	time	1992	Reports monthly.
Manufactured Homes Survey	https://api.census.gov/data/timeseries/eits/mhs?get=data_type_code,time_slot_id,seasonally_adj,category_code,cell_value,error_data&time=1959&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/mhs-api.xml	time	1959	Reports monthly.
Manufactured Housing Survey	https://api.census.gov/data/timeseries/eits/mhs2?get=cell_value,time_slot_id,error_data,category_code&for=US&seasonally_adj&data_type_code&time=1959&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/mhs2-api.xml	for, time	1959	Reports monthly.
Monthly Retail Trade and Food	https://api.census.gov/data/timeseries/eits/mrts?get=cell_value,time_slot_id,error_data,category_code&for=US&seasonally_adj&data_type_code&time=1992&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/mrts-api.xml	time	1992	Reports monthly.

Economic Indicators Time Series API User Guide

Services	t_id,error_data,category_code&for&seasonally_adj&data_type_code&time=1992&key=YOUR_KEY_GOES_HERE	a/datasets/mrts-api.xml			
Manufacturing and Trade Inventories and Sales	https://api.census.gov/data/timeseries/eits/mtis?get=cell_value,time_slot_id,error_data,category_code&for&seasonally_adj&data_type_code&time=1992&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/mtis-api.xml	time	1992	Reports monthly.
Monthly Wholesale Trade: Sales and Inventories	https://api.census.gov/data/timeseries/eits/mwts?get=cell_value,time_slot_id,error_data,category_code&for&seasonally_adj&data_type_code&time=2000&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/mwts-api.xml	time	1992	Reports monthly.
Quarterly Financial Report	https://api.census.gov/data/timeseries/eits/qfr?get=cell_value,time_slot_id,error_data,category_code&for&seasonally_adj&data_type_code&time=2000&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/qfr-api.xml	time	2000	Reports quarterly.
Quarterly Survey of Public Pensions	https://api.census.gov/data/timeseries/eits/qpr?get=cell_value,time_slot_id,error_data,category_code&for&seasonally_adj&data_type_code&time=1968&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/qpr-api.xml	time	1968	Reports quarterly.
Quarterly Services Survey	https://api.census.gov/data/timeseries/eits/qss?get=cell_value,time_slot_id,error_data,category_code&for&seasonally_adj&data_type_code&time=2003&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/qss-api.xml	time	2003	Reports quarterly.
Quarterly Summary of State and Local Taxes	https://api.census.gov/data/timeseries/eits/qtax?get=cell_value,time_slot_id,error_data,category_code&for&seasonally_adj&data_type_code&time=1992&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/qtax-api.xml	time	1992	Reports quarterly. Provides state level data. You cannot specify a state, it simply returns the data for all states.
New Residential Construction	https://api.census.gov/data/timeseries/eits/resconst?get=cell_value,time_slot_id,error_data,category_code&for&seasonally_adj&data_type_code&time=1959&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/resconst-api.xml	time	1959	Reports monthly.
New Homes Sales	https://api.census.gov/data/timeseries/eits/ressales?get=cell_value,time_slot_id,error_data,category_code&for&seasonally_adj&data_type_code&time=1963&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/ressales-api.xml	time	1963	Reports monthly.
Construction Spending	https://api.census.gov/data/timeseries/eits/vip?get=cell_value,time_slot_id,error_data,category_code&for&seasonally_adj&data_type_code&time=2002&key=YOUR_KEY_GOES_HERE	https://www.census.gov/econ/currentdata/datasets/vip-api.xml	time	2002	Reports monthly.

*Key parameter is optional for all datasets

APPENDIX B – PYTHON EXAMPLE CODE



The image shows a Jupyter Notebook interface with the title "Making a Census API Calls Using Python". The notebook contains a single code cell with the following Python code:

```
In [4]: # Add Libraries needed to make an API call
import urllib.request, urllib.parse, urllib.error
import json

# By default because an API key is not needed under 500 request a day.
api_key = False
if api_key is False:
    # If you have an API key you would use it here
    api_key = '4c4d5dcc24fa1b6b04a42d297883ef4934d83c0e'
else:
    api_key = False

# Include the base URL and the Indicator abbreviation for the chosen data set
# Include cell_value and time_slot_id at the end since will never require a value
api_url = 'https://api.census.gov/data/timeseries/eits/qfr?get=cell_value,time_slot_id&'

# Create placeholder for the required parameters
# Set values for error_data, seasonally_adj, category_code, data_type_code, for, and time
error_data = '' # Optional value
seasonally_adj = '' # Optional value
category_code = '' # Optional value
data_type_code = '' # Optional value
for_param = '' # Optional value
time = 'from 2001-Q4 to 2002-Q2' # Required value

# Create a dictionary List of parameters to add to the query string
params = dict()
params['error_data'] = error_data
params['seasonally_adj'] = seasonally_adj
params['category_code'] = category_code
params['data_type_code'] = data_type_code
params['for'] = for_param
params['time'] = time
# Add api key to the end if you have one
if api_key is not False: params['key'] = api_key

# Add parameters to query string
url = api_url + urllib.parse.urlencode(params)

# Print resulting URL before getting API data
print('Retrieving', url)

# Pull API Data
api_data = urllib.request.urlopen(url)
# Read API Data
data = api_data.read().decode()
print('Retrieved', len(data), 'characters')

# Transfer data into JSON format
try:
    js = json.loads(data)
except:
    js = None

# If no JSON is found print results
if not js:
    print('==== Failure To Retrieve ====')
    print(data)

def only_dict(data):
    #Convert json string representation of dictionary to a python dict
    return ast.literal_eval(data)

print(data)
#print(json.dumps(js, indent=4))
```