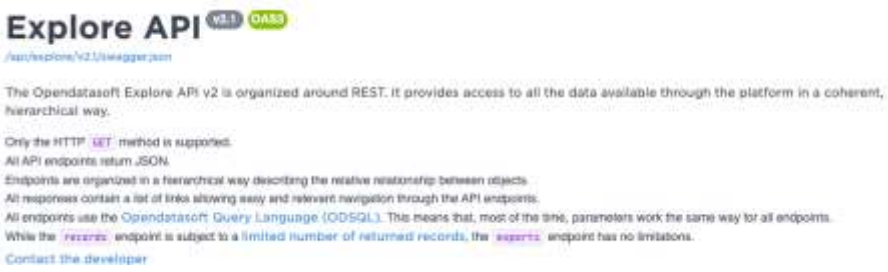
City of Melbourne Open Data | DATA SCIENCE Team

ODSQL & API v2.1 Tutorial 2024

This document explains how to use the API of City of Melbourne Open Data (CoM). Only the HTTP (GET) method is supported, all API endpoints return JSON. Endpoints use ODSQL.

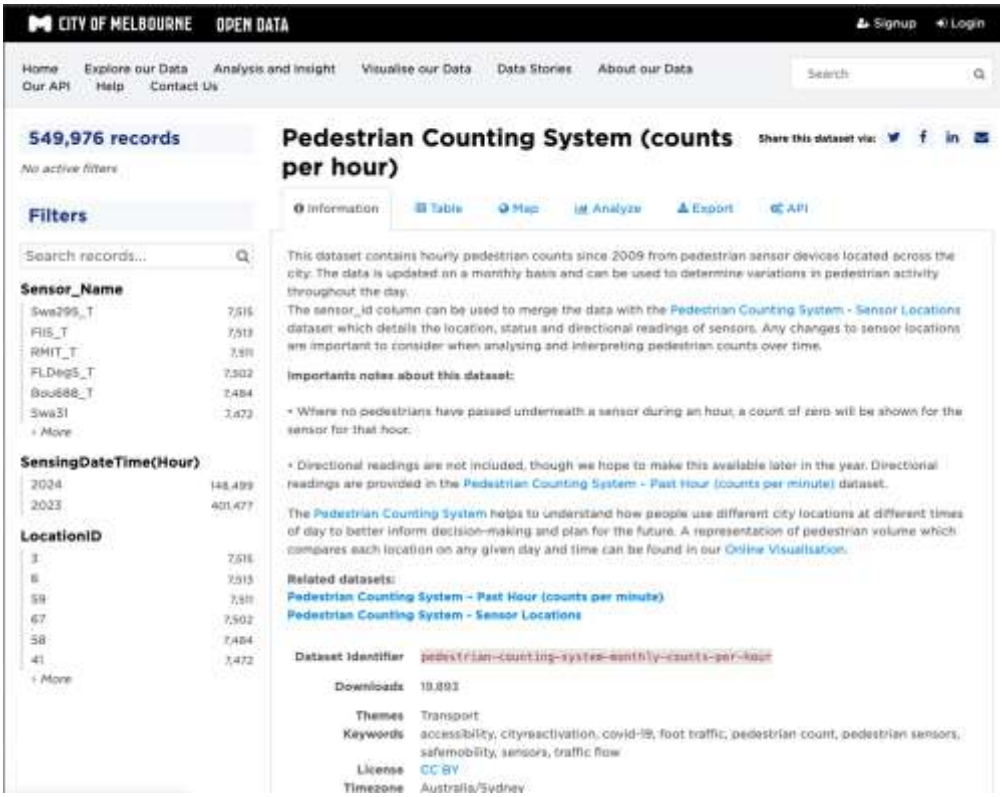


Note

- These instructions only apply to **API v2.1** and the City of Melbourne Open Data.
- `**records**` endpoint is subject to a *limited number of returned records*
- `**exports**` endpoints has no limitations!
- Each organization's API settings are different!

Some parts of the documentation, such as loop acquisition. The government may ban it in subsequent updates, and it does not apply to team' API keys with advanced permissions. For commercial enterprises, loop acquisition is likely to be disabled and may be defaulted by the system as an attack or illegal request.

Explore Datasets



Dataset Pedestrian Counting System (counts per hour)

Appy for Personal API Key

Do not share your API key, keep it secret.

[Instructions on applying for a API key]

API Keys

```
API_KEY  =  os.environ.get('MELBOURNE_API_KEY',  input("Please  enter  your  API  key.  "))
BASE_URL  =  'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/'
```

DO NOT EXPOSE YOUR API KEY – IT IS A SECURITY RISK!

Option 1. Remove API KEY Before Publishing

Risky you could forget!

```
def fetch_data(base_url, dataset, api_key, num_records=99, offset=0):
    all_records = []
    max_offset = 9900

    while True:
        if offset > max_offset:
            break

        filters = f'{dataset}/records?limit={num_records}&offset={offset}'
        url = f'{base_url}{filters}&api_key={api_key}'

        try:
            result = requests.get(url, timeout = 10)
            result.raise_for_status()
            records = result.json().get('results')
        except requests.exceptions.RequestException as e:
            raise Exception(f'API request failed: {e}')
        if records is None:
            break
        all_records.extend(records)
        if len(records) < num_records:
            break

        offset += num_records

    df = pd.DataFrame(all_records)
    return df

BASE_URL = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/'
API_KEY = ''
```

Option 2. Pull API key from Separate File (Locally or GitHub)


```
# Dependencies
import warnings
warnings.filterwarnings("ignore")
pd.set_option('display.max_columns', None)

import requests
import numpy as np
import pandas as pd
```

Cloud or Local IDE (Run notebook/ script)

- To collect API from directory located in Google Collab

```
from google.colab import drive
drive.mount('/content/drive')
with open('/content/drive/My Drive/SIT378/h.txt', 'r') as file:
    api_key = file.read().strip()

import os
api_key = os.getenv(api_key)
```

Mounted at /content/drive

Load Dependencies and load Cloud (Google Collab)

Option 3. Using Google Cloud Secret Manager

```
from google.cloud import secretmanager
client = secretmanager.SecretManagerServiceClient()
name = f"projects/your-project-id/secrets/your-secret-name/versions/latest"
response = client.access_secret_version(request={"name": name})
api_key = response.payload.data.decode("UTF-8")
```

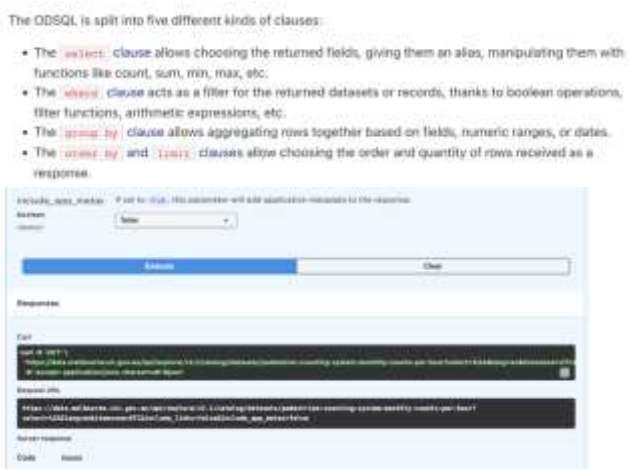
Pre-fill parameters (*Optional: filtering)*

Opendatasoft Query Language (ODSQL) allows you to filter datasets or records and build aggregations.

https://help.opendatasoft.com/apis/ods-explore-v2/#section/Opendatasoft-Query-Language-(ODSQL)

Using the Console you can prefill your parameters and create a GET curl request, you can then use Python HTTP Request (Import requests) library to call API by interacting via HTTP methods.

https://data.melbourne.vic.gov.au/api/explore/v2.1/console



CoM API endpoints: To interact with a data catalog.

- Endpoints allow you to enumerate datasets
- List export formats
- Export data
- List facet values
- Manage individual dataset records

Catalog API

- `GET /catalog/datasets`
- **Purpose** To list all the datasets available in the catalog
  - Used to get an overview of the datasets available in the system
- `GET /catalog/exports`
- **Purpose** To list all export formats that the catalog supports
  - Useful for understanding what formats the data can be exported (CSV, JSON)
- `GET /catalog/exports/{format}`
- **Purpose** To export the entire catalog in a specific format
  - Used when you want to download the entire catalog in one of the supported formats
- `GET /catalog/exports/csv`
- **Purpose** Specifically for exporting the catalog in CSV format.
  - A direct endpoint for exporting data in a common, easily usable format.
- `GET /catalog/exports/dcat{dcat_ap_format}`
- **Purpose** To export the catalog in RDF/XML format using DCAT
  - Exporting data in a format that's suitable for integrating with other data catalogs or systems following the DCAT standard
- `GET /catalog/facets`
- **Purpose** To list all the facet values available in the catalog
  - Facets are used to filter or categorize datasets/ helps understand the categorization
- `GET /catalog/datasets/{dataset_id}`
- **Purpose** To show detailed information about a specific dataset
  - When you need metadata or details about a particular dataset

Dataset API

- `GET /catalog/datasets/{dataset_id}/records`
- **Purpose** To query records within a specific dataset
  - To retrieve the data entries or records from a specific dataset
- `GET /catalog/datasets/{dataset_id}/exports`
- **Purpose** To list the export formats available for a specific dataset
- Understands in what formats you can export the data from this dataset
- `GET /catalog/datasets/{dataset_id}/exports/{format}`
- **Purpose** To export a specific dataset in a specified format
  - To download data from a specific dataset in a particular format
- `GET /catalog/datasets/{dataset_id}/exports/csv`
- **Purpose** To export a specific dataset in CSV format
  - Direct endpoint for exporting dataset data in CSV, a commonly used data format
- `GET /catalog/datasets/{dataset_id}/exports/gpx`
- **Purpose** To export a specific dataset in GPX format
  - Useful for datasets related to geographical data, which GPX format is well-suited for
- `GET /catalog/datasets/{dataset_id}/facets`
- **Purpose** To list the facets for a specific dataset
  - To get an understanding of the different dimensions or categories within a dataset
- `GET /catalog/datasets/{dataset_id}/attachments`
- **Purpose** To list attachments for a specific dataset
  - When datasets have additional files or documents attached, this endpoint lets you enumerate them
- `GET /catalog/datasets/{dataset_id}/records/{record_id}`
- **Purpose** To read a specific record within a dataset
  - To get detailed information about a particular entry or record in a dataset

1. exports API (Endpoint)

**Preferred method by CoM as it has no limitations*

Example: Single Request for CSV file
GET/Catalog/exports

`GET https://data.melbourne.vic.gov.au/api/catalog/datasets/pedestrian-counting-system-monthly-counts-per-hour`

This can take a little while as you are pulling the whole dataset

Feel free to add a progress bar etc

`GET/catalog/exports/catalog/datasets/`

- ODSQL Function Export CSV or json_format
- Read response directly into dataframe
- `response.content.decode('utf-8')` converts binary repsonse into UTF-8 string (encoded)
- Data uses a delimiter (;)

```
# **Preferred Method**: Export Endpoint
import requests
import pandas as pd
from io import StringIO

#Function to collect data
def collect_data(dataset_id):
    base_url = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/'
    #apikey = api_key #use if use datasets API_key permissions
    dataset_id = dataset_id
    format = 'csv'

    url = f'{base_url}{dataset_id}/exports/{format}'
    params = {
        'select': '*',
        'limit': -1,  # all records
        'lang': 'en',
        'timezone': 'UTC',
        #'api_key': apikey  #use if use datasets API_key permissions
    }

    # GET request
    response = requests.get(url, params=params)

    if response.status_code == 200:
        # StringIO to read the CSV data
        url_content = response.content.decode('utf-8')
        dataset = pd.read_csv(StringIO(url_content), delimiter=';')
        return dataset
    else:
        print(f'Request failed with status code {response.status_code}')
```

[7]

Call function to collect specific dataset, check import worked

```
# Set dataset_id to query for the API call dataset name
dataset_id = 'pedestrian-counting-system-monthly-counts-per-hour'
# Save dataset to df varaible
df = collect_data(dataset_id)
# Check number of records in df
print(f'The dataset contains {len(df)} records.')
# View df
df.head(3)
```

```
The dataset contains 549976 records.
```

	sensor_name	timestamp	locationid	direction_1	direction_2	total_of_directions	location
0	SprFli_T	2023-04-24T21:00:00+00:00	75	36	17	53	-37.81515276, 144.97467661
1	SprFli_T	2023-04-25T00:00:00+00:00	75	28	50	78	-37.81515276, 144.97467661
2	SprFli_T	2023-04-25T01:00:00+00:00	75	63	63	126	-37.81515276, 144.97467661

Additional examples exports endpoint

Catalog (API to enumerate datasets) or ***Dataset (API to work on records)***

Example: Catalog API to enumerate datasets
GET/Catalog/datasets

`GET https://data.melbourne.vic.gov.au/api/catalog/datasets`
Limit 10, this can be commented out

```
import requests
url = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets'
params = {
    'select': '*',
    'limit': 10,
    'offset': 0,
    'timezone': 'UTC',
    'include_links': 'false',
    'include_app_metas': 'false'
}
headers = {
    'accept': 'application/json; charset=utf-8'
}

# GET request
response = requests.get(url, headers=headers, params=params)

if response.status_code == 200: # Status code Check
    # Successful
    print(response.json())
else:
    # Error
    print(f'Request failed with status code {response.status_code}')

{'total_count': 228, 'results': [{'visibility': 'domain', 'dataset_id': 'city-of-melbourne-floor-space-forecasts-by-small-area-2020-2040', 'dataset_uid': 'da_wvdwug', 'has_records': Tru
```

Example: Show dataset Information
GET/Catalog/datasets/{dataset_id}

`GET https://data.melbourne.vic.gov.au/api/catalog/datasets/pedestrian-counting-system-monthly-counts-per-hour`

```
[1]
# dataset_id
# https://data.melbourne.vic.gov.au/explore/dataset/pedestrian-counting-system-monthly-counts-per-hour/information/
dataset_id = 'pedestrian-counting-system-monthly-counts-per-hour'
```

```
import requests
url = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/' + dataset_id
# or use full URL
# # https://data.melbourne.vic.gov.au/explore/dataset/pedestrian-counting-system-monthly-counts-per-hour/information/

params = {
    'select': '*',
    'lang': 'en',
    'timezone': 'UTC',
    'include_links': 'false',
    'include_app_metas': 'false'
}
headers = {
    'accept': 'application/json; charset=utf-8'
}

# Make the GET request
response = requests.get(url, headers=headers, params=params)
if response.status_code == 200:
    # Successful
    print(response.json())
else:
    # Error
    print(f'Request failed with status code {response.status_code}')

{'visibility': 'domain', 'dataset_id': 'pedestrian-counting-system-monthly-counts-per-hour', 'dataset_uid': 'da_3k88ps', 'has_records': True, 'features': ['geo', 'analyze', 'timeserie'],
```

Limit parameter controls the number of records or datasets returned in the response.

Example: Check Available export formats for dataset
GET/Catalog/exports

`GET https://data.melbourne.vic.gov.au/api/catalog/datasets/pedestrian-counting-system-monthly-counts-per-hour`

```
import requests
url = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/exports'
params = {
    'select': '*',
    'lang': 'en',
    'timezone': 'UTC',
    'include_links': 'false',
    'include_app_metas': 'false'
}
headers = {
    'accept': 'application/json; charset=utf-8'
}

# Make the GET request
response = requests.get(url, headers=headers, params=params)
if response.status_code == 200:
    # Successful
    print(response.json())
else:
    # Error
    print(f'Request failed with status code {response.status_code}')

', 'href': 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/exports/dcat'}, {'rel': 'rss', 'href': 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/exports/rss'},
```

2. *records* API (Endpoint)

**Not preferred by CoM as it has limitations < 10000 records*

Example: Function to iterate over data chunks using API until max offset reached

{dataset}/records?limit={num_records}&offset={offset}

Records Endpoint

Function `fetch_data` paginates iterates over data in chunks (num_records and offset) until all records are retrieved or a maximum offset is reached.

- This endpoint is subjected to a limited number of returned records: <10000

```python
import requests
import pandas as pd
def fetch_data(base_url, dataset, api_key, num_records=99, offset=0):
    all_records = []
    max_offset = 9900

    while True:
        if offset > max_offset:
            break

        filters = f'{dataset}/records?limit={num_records}&offset={offset}'
        url = f'{base_url}{filters}&api_key={api_key}'

        try:
            result = requests.get(url, timeout = 10)
            result.raise_for_status()
            records = result.json().get('results')
        except requests.exceptions.RequestException as e:
            raise Exception(f'API request failed: {e}')
        if records is None:
            break
        all_records.extend(records)
        if len(records) < num_records:
            break

        offset += num_records

    df = pd.DataFrame(all_records)
    return df

BASE_URL = 'https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/datasets/'
API_KEY = api_key
```

```python
# data set name
SENSOR_DATASET = 'on-street-parking-bay-sensors'
df = fetch_data(BASE_URL, SENSOR_DATASET, API_KEY)
df
```

	lastupdated	status_timestamp	zone_number	status_description	kerbsideid	location
0	2023-10-25T01:44:02+00:00	2023-10-25T01:07:05+00:00	7556.0	Present	5730	{'lon': 144.9680525765466, 'lat': -37.81058233...
1	2023-10-25T01:44:02+00:00	2023-10-25T00:57:43+00:00	7556.0	Present	5728	{'lon': 144.9681094607273, 'lat': -37.81058562...
2	2023-10-25T01:44:02+00:00	2023-10-25T01:12:46+00:00	7556.0	Present	5750	{'lon': 144.9681371221672, 'lat': -37.81055767...

Author

Te' Claire 2024.v1