

The Energy Performance Certificates dataset was updated on **21 Jun 2024** and includes certificates issued up to and including **31 May 2024**.

# Non-domestic Energy Performance Certificates API

## TABLE OF CONTENTS

1. [Using Swagger & OpenAPI](#)
2. [Using this API](#)
3. [Non-domestic Search API](#)
  - [Pagination using `search-after` \(\*\*new\*\*\)](#)
  - [Pagination using `from` \(\*\*deprecated\*\*\)](#)
  - [Filtering by address](#)
  - [Filtering by postcode](#)
  - [Filtering by uprn](#)
  - [Filtering by local authority](#)
  - [Filtering by constituency](#)
  - [Filtering by property type](#)
  - [Filtering by floor area](#)
  - [Filtering by energy band](#)
  - [Filtering by lodgement date](#)
  - [Combining Filters](#)
4. [Non-domestic Certificate API](#)
5. [Non-domestic Recommendations API](#)

## 1: Using Swagger & OpenAPI

As of December 2023, we started providing [OpenAPI v3](#) schemas for all our APIs. These schemas document all our published API routes and describe the format of

the HTTP request parameters we support for querying our API.

In addition to publishing the OpenAPI v3 schemas, we also now [provide a Swagger User Interface](#) which can be used to interactively support developers in making requests against the API.

To use this interface to help build and test API requests you will need to first authorise, by clicking the `Authorize` button in the user interface and providing your `Username` ( `rayan.azhari.17@ucl.ac.uk` ) and `Password` (your API key: `7c87f640d9d041f4268114ae5b9acd775dc2b8c7` ).

Note: These OpenAPI schemas do not describe a new API; they simply describe our existing API following the OpenAPI standards. This provides developers with extra tooling support such as our [Swagger interface](#).

If you have any comments on or feedback regarding these additions, please [contact us](#).

## 2: Using this API

You require a registered account to use this API, and all requests to this API must be authenticated to your account using HTTP Basic Auth. To do this we use a Base64-encoded account identifier composed of your email address and api-key.

Your Base64-encoded authentication token is:

```
cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3NzVhYzJiOGI=
```

This token must then be included in the HTTP `Authorization` header on every API request with the `Basic` authentication scheme, for example:

Both of these tokens are used to authenticate you, so you should take measures to ensure you do not share them publicly, e.g by avoiding committing them into public source code repositories, or storing them in the source of a publicly accessible website.

```
Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YVU1Yj1hY2Q3NzVhYzJiOGI=
```


Assuming you have already registered, you should have received an email containing your API key.

This token is generated by Base64-encoding the string

rayan.azhari.17@uc1.ac.uk:7c87f640d9d041f4268114ae5b9acd775dc2b8c7 which is the concatenation of your email address rayan.azhari.17@uc1.ac.uk with the separator : and your api-key 7c87f640d9d041f4268114ae5b9acd775dc2b8c7 .

You may confirm that this works using a simple [cURL](#) request asking for all results as CSV.

```
$ curl -v -X GET -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QkVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz
```



Alternatively, you may want to create a [Python](#) script to initiate the request.

```
import urllib.request
from urllib.parse import urlencode

token = "cmF5YW4uYXpoYXJpLjE3QkVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz"

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

with urllib.request.urlopen(urllib.request.Request(
    'https://epc.opendatacommunities.org/api/v1/non-domestic/search', headers=headers))
    response_body = response.read()
    print(response_body.decode())
```



**NOTE:** You should take care to move the secret/api-key outside of this script and ensure it is securely stored and injected into the script from elsewhere. Safe credential management is your responsibility, the scripts provided here are only intended as an easy starting point.

Unless otherwise stated each of our API's support the following `Accept` headers:

Response Format	Accept Header
CSV	text/csv

## Response Format

Excel (pre 2007)

Excel (post 2007)

JSON

Zip (certificates.csv &  
recommendations.csv)

## Accept Header

application/vnd.ms-excel

application/vnd.openxmlformats-  
officedocument.spreadsheetml.sheet

application/json

application/zip

# 3: Non-domestic Search API

This API can be called by making an authenticated HTTP `GET` request with an appropriate `Accept` header to the API endpoint:

```
https://epc.opendatacommunities.org/api/v1/non-domestic/search
```

Descriptions of all the columns and fields found in the returned data can be found in the [glossary](#).

This search API follows a subtractive search model, where without any additional filter parameters it will return a paginated result set containing all non-domestic certificates.

## Pagination using `search-after` (new)

This method of pagination should be preferred over the [from pagination method](#) as the total number of records that can be retrieved is not limited to 10,000.

Without any additional query parameters the API will return the first page of all the non-domestic certificates, with a default page size of 25 results. This can be changed by specifying the `size` query parameter up to a maximum of 5,000.

The `search-after` query parameter is used to request further pages of results. The `search-after` value for the next page of results is given in the `X-Next-Search-After` response header, allowing the full set of results to be enumerated. The full request path for the next page is also given in a `Link` response header. Note that the next `search-after` value is not given when there are no results found.

For example, retrieving the first 2,000 results using a `size` of 1,000 using two requests, where the second request uses the `search-after` value from the

response headers of the first request:

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy
...
< Link: /api/v1/non-domestic/search?size=1000&search-after=0123456789abcdef; rel="next"
< X-Next-Search-After: 0123456789abcdef
...
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy
```

## Full search-after example script

Below is a Python script to collect all search results for an example query with more than 5,000 results.

```
import urllib.request
from urllib.parse import urlencode

# Page size (max 5000)
query_size = 5000
# Output file name
output_file = 'output.csv'

# Base url and example query parameters
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/search'
query_params = {'size': query_size}

# Set up authentication
headers = {
    'Accept': 'text/csv',
    'Authorization': 'Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWNh
}

# Keep track of whether we have made at least one request for CSV headers and search-after
first_request = True
# Keep track of search-after from previous request
search_after = None

# Loop over entries in query blocks of up to 5000 to write all the data into a file
with open(output_file, 'w') as file:
    # Perform at least one request; if there's no search_after, there are no further requests
    while search_after != None or first_request:
        # Only set search-after if this isn't the first request
        if not first_request:
            query_params["search-after"] = search_after
```

```
# Set parameters for this query
encoded_params = urlencode(query_params)
full_url = f"{base_url}?{encoded_params}"

# Now make request and extract the data and next search_after
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers))
    response_body = response.read()
    body = response_body.decode()
    search_after = response.getheader('X-Next-Search-After')

# For CSV data, only keep the header row from the first response
if not first_request and body != "":
    body = body.split("\n", 1)[1]

# Write received data
file.write(body)

first_request = False
```

## Pagination using `from` (deprecated)

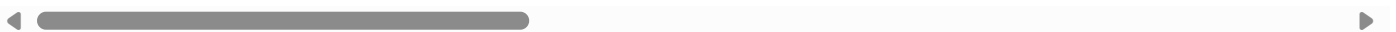
**Warning:** This method of pagination has been deprecated as of December 2023. It is recommended that all users switch to [pagination using the `search-after` method](#).

Without any additional query parameters the API will return the first page of of all the non-domestic certificates, with a default page size of 25 results. This can be changed by specifying the `size` query parameter up to a maximum of 5,000.

Additional pages can be requested by changing the pagination offset with the `from` query string parameter.

For example we can change the page size to `1000` and request the second page of results by setting our `from` parameter to `1000` also:

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy
```



Or in python do:

```
import urllib.request
from urllib.parse import urlencode
```

```

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/search'
query_params = { 'size': 1000, 'from': 1000 }

# Encode query parameters
encoded_params = urlencode(query_params)

# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as resp:
    response_body = response.read()
    print(response_body.decode())

```

Note: This API is only designed to support result sets of up to 10,000 records at a time, with a maximum page size of 5,000. This means you cannot use this method of pagination to query more than 10,000 results for a single search. If you require more than 10,000 records, please use the [search-after pagination method](#), or alternatively you can also download the full zip and work offline.

## Filtering by address

You can filter certificates by address by appending the `address` HTTP query-string parameter to the URL string.

For example we can filter for certificates that have the address `liverpool` :

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy
```

Or in python do:

```

import urllib.request
from urllib.parse import urlencode

```

```

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/search'
query_params = {'address': 'liverpool'}

# Encode query parameters
encoded_params = urlencode(query_params)

# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as resp:
    response_body = response.read()
    print(response_body.decode())

```

The parameter codes supported for this search filter and their meanings are described in the table below:

Parameter/Code	Description
address	An arbitrary search string such as 'liverpool road'

## Filtering by postcode

You can filter certificates by postcode by appending the `postcode` HTTP query-string parameter to the URL string.

For example we can filter for certificates that have the postcode `m1` :

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz" https://epc.opendatacommunities.org/api/v1/non-domestic/search?postcode=m1
```

Or in python do:

```

import urllib.request
from urllib.parse import urlencode

```



```

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/search'
query_params = {'postcode': 'M1'}

# Encode query parameters
encoded_params = urlencode(query_params)

# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as response:
    response_body = response.read()
    print(response_body.decode())

```

The parameter codes supported for this search filter and their meanings are described in the table below:

Parameter/Code	Description
postcode	An arbitrary postcode or prefix string of e.g. 'P', 'PR', 'PR8' or 'PR82EG'

## Filtering by uprn

You can filter certificates by uprn by appending the `uprn` HTTP query-string parameter to the URL string.

You can supply multiple `uprn` parameters if you wish to perform a union search, for example we can filter for certificates that have the values `10094984456` or `5048636` :

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy
```

Or in python do:

```

import urllib.request
from urllib.parse import urlencode

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/search'
query_params = {'uprn': '10094984456', 'uprn': '5048636'}

# Encode query parameters
encoded_params = urlencode(query_params)

# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as resp:
    response_body = response.read()
    print(response_body.decode())

```

The parameter codes supported for this search filter and their meanings are described in the table below:

Parameter/Code	Description
UPRN	Unique Property Reference Number for a building, e.g. 100023336956

## Filtering by local authority

You can filter certificates by local authority by appending the `local-authority` HTTP query-string parameter to the URL string.

For example we can filter for certificates that have the local authority `E07000044` :

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy
```

Or in python do:

```

import urllib.request
from urllib.parse import urlencode

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/search'
query_params = {'local-authority': 'E07000044'}

# Encode query parameters
encoded_params = urlencode(query_params)

# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as response:
    response_body = response.read()
    print(response_body.decode())

```

The parameter codes supported for this search filter and their meanings are described in the table below:

Parameter/Code	Description
E07000223	Adur
E07000026	Allerdale
E07000032	Amber Valley
E07000224	Arun
E07000170	Ashfield
E07000105	Ashford
E07000004	Aylesbury Vale
E07000200	Babergh
E09000002	Barking and Dagenham
E09000003	Barnet
E08000016	Barnsley

Parameter/Code	Description
E07000027	Barrow-in-Furness
E07000066	Basildon
E07000084	Basingstoke and Deane
E07000171	Bassetlaw
E06000022	Bath and North East Somerset
E06000055	Bedford
E09000004	Bexley
E08000025	Birmingham
E07000129	Blaby
E06000008	Blackburn with Darwen
E06000009	Blackpool
W06000019	Blaenau Gwent
E07000033	Bolsover
E08000001	Bolton
E07000136	Boston
E06000058	Bournemouth, Christchurch and Poole
E06000036	Bracknell Forest
E08000032	Bradford
E07000067	Braintree
E07000143	Breckland
E09000005	Brent
E07000068	Brentwood
W06000013	Bridgend
E06000043	Brighton and Hove
E06000023	Bristol, City of
E07000144	Broadland
E09000006	Bromley
E07000234	Bromsgrove
E07000095	Broxbourne
E07000172	Broxtowe
E06000060	Buckinghamshire
E07000117	Burnley
E08000002	Bury

<b>Parameter/Code</b>	<b>Description</b>
W06000018	Caerphilly
E08000033	Calderdale
E07000008	Cambridge
E09000007	Camden
E07000192	Cannock Chase
E07000106	Canterbury
W06000015	Cardiff
E07000028	Carlisle
W06000010	Carmarthenshire
E07000069	Castle Point
E06000056	Central Bedfordshire
W06000008	Ceredigion
E07000130	Charnwood
E07000070	Chelmsford
E07000078	Cheltenham
E07000177	Cherwell
E06000049	Cheshire East
E06000050	Cheshire West and Chester
E07000034	Chesterfield
E07000225	Chichester
E07000005	Chiltern
E07000118	Chorley
E09000001	City of London
E07000071	Colchester
W06000003	Conwy
E07000029	Copeland
E07000150	Corby
E06000052	Cornwall
E07000079	Cotswold
E06000047	County Durham
E08000026	Coventry
E07000163	Craven
E07000226	Crawley

<b>Parameter/Code</b>	<b>Description</b>
E09000008	Croydon
E06000063	Cumberland
E07000096	Dacorum
E06000005	Darlington
E07000107	Dartford
E07000151	Daventry
W06000004	Denbighshire
E06000015	Derby
E07000035	Derbyshire Dales
E08000017	Doncaster
E06000059	Dorset
E07000108	Dover
E08000027	Dudley
E09000009	Ealing
E07000009	East Cambridgeshire
E07000040	East Devon
E07000085	East Hampshire
E07000242	East Hertfordshire
E07000137	East Lindsey
E07000152	East Northamptonshire
E06000011	East Riding of Yorkshire
E07000193	East Staffordshire
E07000244	East Suffolk
E07000061	Eastbourne
E07000086	Eastleigh
E07000030	Eden
E07000207	Elmbridge
E09000010	Enfield
E07000072	Epping Forest
E07000208	Epsom and Ewell
E07000036	Erewash
E07000041	Exeter
E07000087	Fareham

<b>Parameter/Code</b>	<b>Description</b>
E07000010	Fenland
W06000005	Flintshire
E07000112	Folkestone and Hythe
E07000080	Forest of Dean
E07000119	Fylde
E08000037	Gateshead
E07000173	Gedling
E07000081	Gloucester
E07000088	Gosport
E07000109	Gravesham
E07000145	Great Yarmouth
E09000011	Greenwich
E07000209	Guildford
W06000002	Gwynedd
E09000012	Hackney
E06000006	Halton
E07000164	Hambleton
E09000013	Hammersmith and Fulham
E07000131	Harborough
E09000014	Haringey
E07000073	Harlow
E07000165	Harrogate
E09000015	Harrow
E07000089	Hart
E06000001	Hartlepool
E07000062	Hastings
E07000090	Havant
E09000016	Havering
E06000019	Herefordshire, County of
E07000098	Hertsmere
E07000037	High Peak
E09000017	Hillingdon
E07000132	Hinckley and Bosworth

<b>Parameter/Code</b>	<b>Description</b>
E07000227	Horsham
E09000018	Hounslow
E07000011	Huntingdonshire
E07000120	Hyndburn
E07000202	Ipswich
W06000001	Isle of Anglesey
E06000046	Isle of Wight
E06000053	Isles of Scilly
E09000019	Islington
E09000020	Kensington and Chelsea
E07000153	Kettering
E07000146	King's Lynn and West Norfolk
E06000010	Kingston upon Hull, City of
E09000021	Kingston upon Thames
E08000034	Kirklees
E08000011	Knowsley
E09000022	Lambeth
E07000121	Lancaster
E08000035	Leeds
E06000016	Leicester
E07000063	Lewes
E09000023	Lewisham
E07000194	Lichfield
E07000138	Lincoln
E08000012	Liverpool
E06000032	Luton
E07000110	Maidstone
E07000074	Maldon
E07000235	Malvern Hills
E08000003	Manchester
E07000174	Mansfield
E06000035	Medway
E07000133	Melton



<b>Parameter/Code</b>	<b>Description</b>
E07000187	Mendip
W06000024	Merthyr Tydfil
E09000024	Merton
E07000042	Mid Devon
E07000203	Mid Suffolk
E07000228	Mid Sussex
E06000002	Middlesbrough
E06000042	Milton Keynes
E07000210	Mole Valley
W06000021	Monmouthshire
W06000012	Neath Port Talbot
E07000091	New Forest
E07000175	Newark and Sherwood
E08000021	Newcastle upon Tyne
E07000195	Newcastle-under-Lyme
E09000025	Newham
W06000022	Newport
E07000043	North Devon
E07000038	North East Derbyshire
E06000012	North East Lincolnshire
E07000099	North Hertfordshire
E07000139	North Kesteven
E06000013	North Lincolnshire
E07000147	North Norfolk
E06000061	North Northamptonshire
E06000024	North Somerset
E08000022	North Tyneside
E07000218	North Warwickshire
E07000134	North West Leicestershire
E06000065	North Yorkshire
E07000154	Northampton
E06000057	Northumberland
E07000148	Norwich

<b>Parameter/Code</b>	<b>Description</b>
E06000018	Nottingham
E07000219	Nuneaton and Bedworth
E07000135	Oadby and Wigston
E08000004	Oldham
E07000178	Oxford
W06000009	Pembrokeshire
E07000122	Pendle
E06000031	Peterborough
E06000026	Plymouth
E06000044	Portsmouth
W06000023	Powys
E07000123	Preston
E06000038	Reading
E09000026	Redbridge
E06000003	Redcar and Cleveland
E07000236	Redditch
E07000211	Reigate and Banstead
W06000016	Rhondda Cynon Taf
E07000124	Ribble Valley
E09000027	Richmond upon Thames
E07000166	Richmondshire
E08000005	Rochdale
E07000075	Rochford
E07000125	Rossendale
E07000064	Rother
E08000018	Rotherham
E07000220	Rugby
E07000212	Runnymede
E07000176	Rushcliffe
E07000092	Rushmoor
E06000017	Rutland
E07000167	Ryedale
E08000006	Salford

<b>Parameter/Code</b>	<b>Description</b>
E08000028	Sandwell
E07000168	Scarborough
E07000188	Sedgemoor
E08000014	Sefton
E07000169	Selby
E07000111	Sevenoaks
E08000019	Sheffield
E06000051	Shropshire
E06000039	Slough
E08000029	Solihull
E06000066	Somerset
E07000246	Somerset West and Taunton
E07000006	South Bucks
E07000012	South Cambridgeshire
E07000039	South Derbyshire
E06000025	South Gloucestershire
E07000044	South Hams
E07000140	South Holland
E07000141	South Kesteven
E07000031	South Lakeland
E07000149	South Norfolk
E07000155	South Northamptonshire
E07000179	South Oxfordshire
E07000126	South Ribble
E07000189	South Somerset
E07000196	South Staffordshire
E08000023	South Tyneside
E06000045	Southampton
E06000033	Southend-on-Sea
E09000028	Southwark
E07000213	Spelthorne
E07000240	St Albans
E08000013	St. Helens

<b>Parameter/Code</b>	<b>Description</b>
E07000197	Stafford
E07000198	Staffordshire Moorlands
E07000243	Stevenage
E08000007	Stockport
E06000004	Stockton-on-Tees
E06000021	Stoke-on-Trent
E07000221	Stratford-on-Avon
E07000082	Stroud
E08000024	Sunderland
E07000214	Surrey Heath
E09000029	Sutton
E07000113	Swale
W06000011	Swansea
E06000030	Swindon
E08000008	Tameside
E07000199	Tamworth
E07000215	Tandridge
E07000045	Teignbridge
E06000020	Telford and Wrekin
E07000076	Tendring
E07000093	Test Valley
E07000083	Tewkesbury
E07000114	Thanet
E07000102	Three Rivers
E06000034	Thurrock
E07000115	Tonbridge and Malling
E06000027	Torbay
W06000020	Torfaen
E07000046	Torridge
E09000030	Tower Hamlets
E08000009	Trafford
E07000116	Tunbridge Wells
E07000077	Uttlesford

<b>Parameter/Code</b>	<b>Description</b>
W06000014	Vale of Glamorgan
E07000180	Vale of White Horse
E08000036	Wakefield
E08000030	Walsall
E09000031	Waltham Forest
E09000032	Wandsworth
E06000007	Warrington
E07000222	Warwick
E07000103	Watford
E07000216	Waverley
E07000065	Wealden
E07000156	Wellingborough
E07000241	Welwyn Hatfield
E06000037	West Berkshire
E07000047	West Devon
E07000127	West Lancashire
E07000142	West Lindsey
E06000062	West Northamptonshire
E07000181	West Oxfordshire
E07000245	West Suffolk
E09000033	Westminster
E06000064	Westmorland and Furness
E08000010	Wigan
E06000054	Wiltshire
E07000094	Winchester
E06000040	Windsor and Maidenhead
E08000015	Wirral
E07000217	Woking
E06000041	Wokingham
E08000031	Wolverhampton
E07000237	Worcester
E07000229	Worthing
W06000006	Wrexham

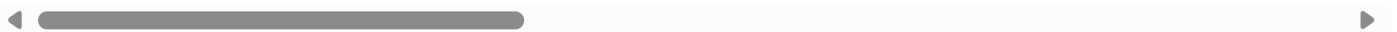
Parameter/Code	Description
E07000238	Wychavon
E07000007	Wycombe
E07000128	Wyre
E07000239	Wyre Forest
E06000014	York
_unknown_local_authority	[Unknown Local Authority]

## Filtering by constituency

You can filter certificates by constituency by appending the `constituency` HTTP query-string parameter to the URL string.

For example we can filter for certificates that have the constituency `E14000645` :

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz
```



Or in python do:

```
import urllib.request
from urllib.parse import urlencode

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz"

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/search'
query_params = {'constituency': 'E14000645'}

# Encode query parameters
encoded_params = urlencode(query_params)

# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as resp
```

```
response_body = response.read()
print(response_body.decode())
```

The parameter codes supported for this search filter and their meanings are described in the table below:

Parameter/Code	Description
W07000049	Aberavon
W07000058	Aberconwy
E14000530	Aldershot
E14000531	Aldridge-Brownhills
E14000532	Altrincham and Sale West
W07000043	Alyn and Deeside
E14000533	Amber Valley
W07000057	Arfon
E14000534	Arundel and South Downs
E14000535	Ashfield
E14000536	Ashford
E14000537	Ashton-under-Lyne
E14000538	Aylesbury
E14000539	Banbury
E14000540	Barking
E14000541	Barnsley Central
E14000542	Barnsley East
E14000543	Barrow and Furness
E14000544	Basildon and Billericay
E14000545	Basingstoke
E14000546	Bassetlaw
E14000547	Bath
E14000548	Batley and Spen
E14000549	Battersea
E14000550	Beaconsfield
E14000551	Beckenham
E14000552	Bedford
E14000553	Bermondsey and Old Southwark

<b>Parameter/Code</b>	<b>Description</b>
E14000554	Berwick-upon-Tweed
E14000555	Bethnal Green and Bow
E14000556	Beverley and Holderness
E14000557	Bexhill and Battle
E14000558	Bexleyheath and Crayford
E14000559	Birkenhead
E14000560	Birmingham, Edgbaston
E14000561	Birmingham, Erdington
E14000562	Birmingham, Hall Green
E14000563	Birmingham, Hodge Hill
E14000564	Birmingham, Ladywood
E14000565	Birmingham, Northfield
E14000566	Birmingham, Perry Barr
E14000567	Birmingham, Selly Oak
E14000568	Birmingham, Yardley
E14000569	Bishop Auckland
E14000570	Blackburn
E14000571	Blackley and Broughton
E14000572	Blackpool North and Cleveleys
E14000573	Blackpool South
W07000072	Blaenau Gwent
E14000574	Blaydon
E14000575	Blyth Valley
E14000576	Bognor Regis and Littlehampton
E14000577	Bolsover
E14000578	Bolton North East
E14000579	Bolton South East
E14000580	Bolton West
E14000581	Bootle
E14000582	Boston and Skegness
E14000583	Bosworth
E14000584	Bournemouth East
E14000585	Bournemouth West



<b>Parameter/Code</b>	<b>Description</b>
E14000586	Bracknell
E14000587	Bradford East
E14000588	Bradford South
E14000589	Bradford West
E14000590	Braintree
W07000068	Brecon and Radnorshire
E14000591	Brent Central
E14000592	Brent North
E14000593	Brentford and Isleworth
E14000594	Brentwood and Ongar
W07000073	Bridgend
E14000595	Bridgwater and West Somerset
E14000596	Brigg and Goole
E14000597	Brighton, Kemptown
E14000598	Brighton, Pavilion
E14000599	Bristol East
E14000600	Bristol North West
E14000601	Bristol South
E14000602	Bristol West
E14000603	Broadland
E14000604	Bromley and Chislehurst
E14000605	Bromsgrove
E14000606	Broxbourne
E14000607	Broxtowe
E14000608	Buckingham
E14000609	Burnley
E14000610	Burton
E14000611	Bury North
E14000612	Bury South
E14000613	Bury St Edmunds
W07000076	Caerphilly
E14000614	Calder Valley
E14000615	Camberwell and Peckham

<b>Parameter/Code</b>	<b>Description</b>
E14000616	Camborne and Redruth
E14000617	Cambridge
E14000618	Cannock Chase
E14000619	Canterbury
W07000050	Cardiff Central
W07000051	Cardiff North
W07000080	Cardiff South and Penarth
W07000079	Cardiff West
E14000620	Carlisle
W07000067	Carmarthen East and Dinefwr
W07000066	Carmarthen West and South Pembrokeshire
E14000621	Carshalton and Wallington
E14000622	Castle Point
E14000623	Central Devon
E14000624	Central Suffolk and North Ipswich
W07000064	Ceredigion
E14000625	Charnwood
E14000626	Chatham and Aylesford
E14000627	Cheadle
E14000628	Chelmsford
E14000629	Chelsea and Fulham
E14000630	Cheltenham
E14000631	Chesham and Amersham
E14000632	Chesterfield
E14000633	Chichester
E14000634	Chingford and Woodford Green
E14000635	Chippenham
E14000636	Chipping Barnet
E14000637	Chorley
E14000638	Christchurch
E14000639	Cities of London and Westminster
E14000640	City of Chester
E14000641	City of Durham

<b>Parameter/Code</b>	<b>Description</b>
E14000642	Clacton
E14000643	Cleethorpes
W07000062	Clwyd South
W07000059	Clwyd West
E14000644	Colchester
E14000645	Colne Valley
E14000646	Congleton
E14000647	Copeland
E14000648	Corby
E14000649	Coventry North East
E14000650	Coventry North West
E14000651	Coventry South
E14000652	Crawley
E14000653	Crewe and Nantwich
E14000654	Croydon Central
E14000655	Croydon North
E14000656	Croydon South
W07000070	Cynon Valley
E14000657	Dagenham and Rainham
E14000658	Darlington
E14000659	Dartford
E14000660	Daventry
W07000042	Delyn
E14000661	Denton and Reddish
E14000662	Derby North
E14000663	Derby South
E14000664	Derbyshire Dales
E14000665	Devizes
E14000666	Dewsbury
E14000667	Don Valley
E14000668	Doncaster Central
E14000669	Doncaster North
E14000670	Dover

<b>Parameter/Code</b>	<b>Description</b>
E14000671	Dudley North
E14000672	Dudley South
E14000673	Dulwich and West Norwood
W07000061	Dwyfor Meirionnydd
E14000674	Ealing Central and Acton
E14000675	Ealing North
E14000676	Ealing, Southall
E14000677	Easington
E14000678	East Devon
E14000679	East Ham
E14000680	East Hampshire
E14000681	East Surrey
E14000682	East Worthing and Shoreham
E14000683	East Yorkshire
E14000684	Eastbourne
E14000685	Eastleigh
E14000686	Eddisbury
E14000687	Edmonton
E14000688	Ellesmere Port and Neston
E14000689	Elmet and Rothwell
E14000690	Eltham
E14000691	Enfield North
E14000692	Enfield, Southgate
E14000693	Epping Forest
E14000694	Epsom and Ewell
E14000695	Erewash
E14000696	Erith and Thamesmead
E14000697	Esher and Walton
E14000698	Exeter
E14000699	Fareham
E14000700	Faversham and Mid Kent
E14000701	Feltham and Heston
E14000702	Filton and Bradley Stoke

<b>Parameter/Code</b>	<b>Description</b>
E14000703	Finchley and Golders Green
E14000704	Folkestone and Hythe
E14000705	Forest of Dean
E14000706	Fylde
E14000707	Gainsborough
E14000708	Garston and Halewood
E14000709	Gateshead
E14000710	Gedling
E14000711	Gillingham and Rainham
E14000712	Gloucester
E14000713	Gosport
W07000046	Gower
E14000714	Grantham and Stamford
E14000715	Gravesham
E14000716	Great Grimsby
E14000717	Great Yarmouth
E14000718	Greenwich and Woolwich
E14000719	Guildford
E14000720	Hackney North and Stoke Newington
E14000721	Hackney South and Shoreditch
E14000722	Halesowen and Rowley Regis
E14000723	Halifax
E14000724	Haltemprice and Howden
E14000725	Halton
E14000726	Hammersmith
E14000727	Hampstead and Kilburn
E14000728	Harborough
E14000729	Harlow
E14000730	Harrogate and Knaresborough
E14000731	Harrow East
E14000732	Harrow West
E14000733	Hartlepool
E14000734	Harwich and North Essex

<b>Parameter/Code</b>	<b>Description</b>
E14000735	Hastings and Rye
E14000736	Havant
E14000737	Hayes and Harlington
E14000738	Hazel Grove
E14000739	Hemel Hempstead
E14000740	Hemsworth
E14000741	Hendon
E14000742	Henley
E14000743	Hereford and South Herefordshire
E14000744	Hertford and Stortford
E14000745	Hertsmere
E14000746	Hexham
E14000747	Heywood and Middleton
E14000748	High Peak
E14000749	Hitchin and Harpenden
E14000750	Holborn and St Pancras
E14000751	Hornchurch and Upminster
E14000752	Hornsey and Wood Green
E14000753	Horsham
E14000754	Houghton and Sunderland South
E14000755	Hove
E14000756	Huddersfield
E14000757	Huntingdon
E14000758	Hyndburn
E14000759	Ilford North
E14000760	Ilford South
E14000761	Ipswich
E14000762	Isle of Wight
E14000763	Islington North
E14000764	Islington South and Finsbury
W07000077	Islwyn
E14000765	Jarrow
E14000766	Keighley

<b>Parameter/Code</b>	<b>Description</b>
E14000767	Kenilworth and Southam
E14000768	Kensington
E14000769	Kettering
E14000770	Kingston and Surbiton
E14000771	Kingston upon Hull East
E14000772	Kingston upon Hull North
E14000773	Kingston upon Hull West and Hessle
E14000774	Kingswood
E14000775	Knowsley
E14000776	Lancaster and Fleetwood
E14000777	Leeds Central
E14000778	Leeds East
E14000779	Leeds North East
E14000780	Leeds North West
E14000781	Leeds West
E14000782	Leicester East
E14000783	Leicester South
E14000784	Leicester West
E14000785	Leigh
E14000786	Lewes
E14000787	Lewisham East
E14000788	Lewisham West and Penge
E14000789	Lewisham, Deptford
E14000790	Leyton and Wanstead
E14000791	Lichfield
E14000792	Lincoln
E14000793	Liverpool, Riverside
E14000794	Liverpool, Walton
E14000795	Liverpool, Wavertree
E14000796	Liverpool, West Derby
W07000045	Llanelli
E14000797	Loughborough
E14000798	Louth and Horncastle

<b>Parameter/Code</b>	<b>Description</b>
E14000799	Ludlow
E14000800	Luton North
E14000801	Luton South
E14000802	Macclesfield
E14000803	Maidenhead
E14000804	Maidstone and The Weald
E14000805	Makerfield
E14000806	Maldon
E14000807	Manchester Central
E14000808	Manchester, Gorton
E14000809	Manchester, Withington
E14000810	Mansfield
E14000811	Meon Valley
E14000812	Meriden
W07000071	Merthyr Tydfil and Rhymney
E14000813	Mid Bedfordshire
E14000814	Mid Derbyshire
E14000815	Mid Dorset and North Poole
E14000816	Mid Norfolk
E14000817	Mid Sussex
E14000818	Mid Worcestershire
E14000819	Middlesbrough
E14000820	Middlesbrough South and East Cleveland
E14000821	Milton Keynes North
E14000822	Milton Keynes South
E14000823	Mitcham and Morden
E14000824	Mole Valley
W07000054	Monmouth
W07000063	Montgomeryshire
E14000825	Morecambe and Lunesdale
E14000826	Morley and Outwood
W07000069	Neath
E14000827	New Forest East



<b>Parameter/Code</b>	<b>Description</b>
E14000828	New Forest West
E14000829	Newark
E14000830	Newbury
E14000831	Newcastle upon Tyne Central
E14000832	Newcastle upon Tyne East
E14000833	Newcastle upon Tyne North
E14000834	Newcastle-under-Lyme
W07000055	Newport East
W07000056	Newport West
E14000835	Newton Abbot
E14000836	Normanton, Pontefract and Castleford
E14000837	North Cornwall
E14000838	North Devon
E14000839	North Dorset
E14000840	North Durham
E14000841	North East Bedfordshire
E14000842	North East Cambridgeshire
E14000843	North East Derbyshire
E14000844	North East Hampshire
E14000845	North East Hertfordshire
E14000846	North East Somerset
E14000847	North Herefordshire
E14000848	North Norfolk
E14000849	North Shropshire
E14000850	North Somerset
E14000851	North Swindon
E14000852	North Thanet
E14000853	North Tyneside
E14000854	North Warwickshire
E14000855	North West Cambridgeshire
E14000856	North West Durham
E14000857	North West Hampshire
E14000858	North West Leicestershire

<b>Parameter/Code</b>	<b>Description</b>
E14000859	North West Norfolk
E14000860	North Wiltshire
E14000861	Northampton North
E14000862	Northampton South
E14000863	Norwich North
E14000864	Norwich South
E14000865	Nottingham East
E14000866	Nottingham North
E14000867	Nottingham South
E14000868	Nuneaton
W07000074	Ogmore
E14000869	Old Bexley and Sidcup
E14000870	Oldham East and Saddleworth
E14000871	Oldham West and Royton
E14000872	Orpington
E14000873	Oxford East
E14000874	Oxford West and Abingdon
E14000875	Pendle
E14000876	Penistone and Stocksbridge
E14000877	Penrith and The Border
E14000878	Peterborough
E14000879	Plymouth, Moor View
E14000880	Plymouth, Sutton and Devonport
W07000075	Pontypridd
E14000881	Poole
E14000882	Poplar and Limehouse
E14000883	Portsmouth North
E14000884	Portsmouth South
W07000065	Preseli Pembrokeshire
E14000885	Preston
E14000886	Pudsey
E14000887	Putney
E14000888	Rayleigh and Wickford

<b>Parameter/Code</b>	<b>Description</b>
E14000889	Reading East
E14000890	Reading West
E14000891	Redcar
E14000892	Redditch
E14000893	Reigate
W07000052	Rhondda
E14000894	Ribble Valley
E14000895	Richmond (Yorks)
E14000896	Richmond Park
E14000897	Rochdale
E14000898	Rochester and Strood
E14000899	Rochford and Southend East
E14000900	Romford
E14000901	Romsey and Southampton North
E14000902	Rossendale and Darwen
E14000903	Rother Valley
E14000904	Rotherham
E14000905	Rugby
E14000906	Ruislip, Northwood and Pinner
E14000907	Runnymede and Weybridge
E14000908	Rushcliffe
E14000909	Rutland and Melton
E14000910	Saffron Walden
E14000911	Salford and Eccles
E14000912	Salisbury
E14000913	Scarborough and Whitby
E14000914	Scunthorpe
E14000915	Sedgefield
E14000916	Sefton Central
E14000917	Selby and Ainsty
E14000918	Sevenoaks
E14000919	Sheffield Central
E14000920	Sheffield South East

<b>Parameter/Code</b>	<b>Description</b>
E14000921	Sheffield, Brightside and Hillsborough
E14000922	Sheffield, Hallam
E14000923	Sheffield, Heeley
E14000924	Sherwood
E14000925	Shipley
E14000926	Shrewsbury and Atcham
E14000927	Sittingbourne and Sheppey
E14000928	Skipton and Ripon
E14000929	Sleaford and North Hykeham
E14000930	Slough
E14000931	Solihull
E14000932	Somerton and Frome
E14000933	South Basildon and East Thurrock
E14000934	South Cambridgeshire
E14000935	South Derbyshire
E14000936	South Dorset
E14000937	South East Cambridgeshire
E14000938	South East Cornwall
E14000939	South Holland and The Deepings
E14000940	South Leicestershire
E14000941	South Norfolk
E14000942	South Northamptonshire
E14000943	South Ribble
E14000944	South Shields
E14000945	South Staffordshire
E14000946	South Suffolk
E14000947	South Swindon
E14000948	South Thanet
E14000949	South West Bedfordshire
E14000950	South West Devon
E14000951	South West Hertfordshire
E14000952	South West Norfolk
E14000953	South West Surrey

<b>Parameter/Code</b>	<b>Description</b>
E14000954	South West Wiltshire
E14000955	Southampton, Itchen
E14000956	Southampton, Test
E14000957	Southend West
E14000958	Southport
E14000959	Spelthorne
E14000960	St Albans
E14000961	St Austell and Newquay
E14000962	St Helens North
E14000963	St Helens South and Whiston
E14000964	St Ives
E14000965	Stafford
E14000966	Staffordshire Moorlands
E14000967	Stalybridge and Hyde
E14000968	Stevenage
E14000969	Stockport
E14000970	Stockton North
E14000971	Stockton South
E14000972	Stoke-on-Trent Central
E14000973	Stoke-on-Trent North
E14000974	Stoke-on-Trent South
E14000975	Stone
E14000976	Stourbridge
E14000977	Stratford-on-Avon
E14000978	Streatham
E14000979	Stretford and Urmston
E14000980	Stroud
E14000981	Suffolk Coastal
E14000982	Sunderland Central
E14000983	Surrey Heath
E14000985	Sutton Coldfield
E14000984	Sutton and Cheam
W07000048	Swansea East

<b>Parameter/Code</b>	<b>Description</b>
W07000047	Swansea West
E14000986	Tamworth
E14000987	Tatton
E14000988	Taunton Deane
E14000989	Telford
E14000990	Tewkesbury
E14000991	The Cotswolds
E14000992	The Wrekin
E14000993	Thirsk and Malton
E14000994	Thornbury and Yate
E14000995	Thurrock
E14000996	Tiverton and Honiton
E14000997	Tonbridge and Malling
E14000998	Tooting
E14000999	Torbay
W07000053	Torfaen
E14001000	Torridge and West Devon
E14001001	Totnes
E14001002	Tottenham
E14001003	Truro and Falmouth
E14001004	Tunbridge Wells
E14001005	Twickenham
E14001006	Tynemouth
E14001007	Uxbridge and South Ruislip
W07000060	Vale of Clwyd
W07000078	Vale of Glamorgan
E14001008	Vauxhall
E14001009	Wakefield
E14001010	Wallasey
E14001011	Walsall North
E14001012	Walsall South
E14001013	Walthamstow
E14001014	Wansbeck

<b>Parameter/Code</b>	<b>Description</b>
E14001015	Wantage
E14001016	Warley
E14001017	Warrington North
E14001018	Warrington South
E14001019	Warwick and Leamington
E14001020	Washington and Sunderland West
E14001021	Watford
E14001022	Waveney
E14001023	Wealden
E14001024	Weaver Vale
E14001025	Wellingborough
E14001026	Wells
E14001027	Welwyn Hatfield
E14001028	Wentworth and Dearne
E14001029	West Bromwich East
E14001030	West Bromwich West
E14001031	West Dorset
E14001032	West Ham
E14001033	West Lancashire
E14001034	West Suffolk
E14001035	West Worcestershire
E14001036	Westminster North
E14001037	Westmorland and Lonsdale
E14001038	Weston-Super-Mare
E14001039	Wigan
E14001040	Wimbledon
E14001041	Winchester
E14001042	Windsor
E14001043	Wirral South
E14001044	Wirral West
E14001045	Witham
E14001046	Witney
E14001047	Woking

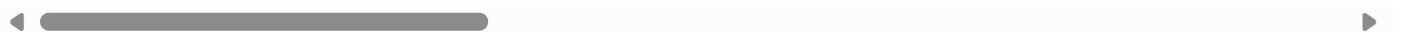
Parameter/Code	Description
E14001048	Wokingham
E14001049	Wolverhampton North East
E14001050	Wolverhampton South East
E14001051	Wolverhampton South West
E14001052	Worcester
E14001053	Workington
E14001054	Worsley and Eccles South
E14001055	Worthing West
W07000044	Wrexham
E14001056	Wycombe
E14001058	Wyre Forest
E14001057	Wyre and Preston North
E14001059	Wythenshawe and Sale East
E14001060	Yeovil
W07000041	Ynys Môn
E14001061	York Central
E14001062	York Outer
_unknown_constituency	[Unknown Constituency]

## Filtering by property type

You can filter certificates by property type by appending the `property-type` HTTP query-string parameter to the URL string.

You can supply multiple `property-type` parameters if you wish to perform a union search, for example we can filter for certificates that have the values `a1-a2` or `a3-a4-a5` :

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYj
```



Or in python do:

```
import urllib.request
from urllib.parse import urlencode
```



```

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/search'
query_params = {'property-type': 'a1-a2', 'property-type': 'a3-a4-a5'}

# Encode query parameters
encoded_params = urlencode(query_params)

# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as resp:
    response_body = response.read()
    print(response_body.decode())

```

The parameter codes supported for this search filter and their meanings are described in the table below:

Parameter/Code	Description
a1-a2	A1/A2 Retail and Financial/Professional services
a3-a4-a5	A3/A4/A5 Restaurant and Cafes/Drinking Establishments and Hot Food takeaways
airport-terminal	Airport terminals
b1-office	B1 Offices and Workshop businesses
b2-to-b7	B2 to B7 General Industrial and Special Industrial Groups
b8-storage	B8 Storage or Distribution
bus-train-seaport-station	Bus station/train station/seaport terminal
c1-hotel	C1 Hotels
c2-res-hospital	C2 Residential Institutions - Hospitals and Care Homes
c2-res-school	C2 Residential Institutions - Residential schools
c2-university	C2 Residential Institutions - Universities and colleges
c2a-secure-res	C2A Secure Residential Institutions
community-day-centre	Community/day centre

Parameter/Code	Description
crown-county-court	Crown and county courts
d1-community	D1 Non-residential Institutions - Community/Day Centre
d1-crown-county-court	D1 Non-residential Institutions - Crown and County Courts
d1-education	D1 Non-residential Institutions - Education
d1-libraries-museum	D1 Non-residential Institutions - Libraries Museums and Galleries
d1-primary-health-care	D1 Non-residential Institutions - Primary Health Care Building
d2-leisure	D2 General Assembly and Leisure plus Night Clubs and Theatres
dwelling	Dwelling
emergency-service	Emergency services
further-education	Further education universities
hospital	Hospital
hotel	Hotel
industrial	Industrial process building
laundrette	Laundrette
libraries-museums-gallery	Libraries/museums/galleries
miscellaneous-24	Miscellaneous 24hr activities
nursing-residential	Nursing residential homes and hostels
office	Office
others-car-park	Others - Car Parks 24 hrs
others-emergency-service	Others - Emergency services
others-misc-24h	Others - Miscellaneous 24hr activities
others-passenger-terminal	Others - Passenger terminals
others-telephone-exchange	Others -Telephone exchanges
others-utility-block	Others - Stand alone utility block
primary-health-care	Primary health care buildings
primary-school	Primary school
prison	Prisons
residential-space	Residential spaces
restaurant-public-house	Restaurant/public house
retail	Retail
retail-warehouse	Retail warehouses

Parameter/Code	Description
secondary-school	Secondary school
social-club	Social clubs
sports-centre-leisure-centre	Sports centre/leisure centre
sports-ground-arena	Sports ground arena
telephone-exchange	Telephone exchanges
theatre-cinema-music	Theatres/cinemas/music halls and auditoria
warehouse-storage	Warehouse and storage
workshop-maintenance-depot	Workshops/maintenance depot

## Filtering by floor area

You can filter certificates by floor area by appending the `floor-area` HTTP query-string parameter to the URL string.

You can supply multiple `floor-area` parameters if you wish to perform a union search, for example we can filter for certificates that have the values `l` or `m`:

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz
```



Or in python do:

```
import urllib.request
from urllib.parse import urlencode

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz"

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/search'
query_params = {'floor-area': 'l', 'floor-area': 'm'}

# Encode query parameters
encoded_params = urlencode(query_params)
```

```
# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as response:
    response_body = response.read()
    print(response_body.decode())
```

The parameter codes supported for this search filter and their meanings are described in the table below:

Parameter/Code	Description
xxxl	
unknown	A floor area recorded as 0
s	A floor area between 1m <sup>2</sup> and 80m <sup>2</sup>
m	A floor area between 80m <sup>2</sup> and 160m <sup>2</sup>
l	A floor area between 160m <sup>2</sup> and 300m <sup>2</sup>
xl	A floor area between 300m <sup>2</sup> and 700m <sup>2</sup>
xxl	A floor area greater than 700m <sup>2</sup>

## Filtering by energy band

You can filter certificates by energy band by appending the `energy-band` HTTP query-string parameter to the URL string.

You can supply multiple `energy-band` parameters if you wish to perform a union search, for example we can filter for certificates that have the values `a+` or `a` :

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz
```

Or in python do:

```
import urllib.request
from urllib.parse import urlencode

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz"

headers = {
```

```

'Accept': 'text/csv',
'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/search'
query_params = {'energy-band': 'a+', 'energy-band': 'a'}

# Encode query parameters
encoded_params = urlencode(query_params)

# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as resp:
    response_body = response.read()
    print(response_body.decode())

```

The parameter codes supported for this search filter and their meanings are described in the table below:

Parameter/Code	Description
a+	A+ rated (net zero)
a	A rated (0-25)
b	B rated (26-50)
c	C rated (51-75)
d	D rated (76-100)
e	E rated (101-125)
f	F rated (126-150)
g	G rated (150+)

## Filtering by lodgement date

You can filter certificates by lodgement date by appending the appropriate date parameter to the HTTP query-string:

Parameter/Code	Description
from-month	A numeric month identifier 1-12, to establish the start of a date range, where 1 is January and 12 is December. If no from-month parameter is supplied 1 (January) is assumed.

Parameter/Code	Description
from-year	A numeric year identifier to establish the start of a date range between 2008 and 2024 e.g. 2015. If no year parameter is supplied 2008 is assumed.
to-month	A numeric month identifier 1-12, to establish the end of a date range, where 1 is January and 12 is December. If no to-month parameter is supplied then 12 is assumed.
to-year	A numeric year identifier between 2008 and 2024 e.g. 2015. If no to-year parameter is supplied then 2024 is assumed.

For example we can filter for certificates that were lodged between May 2015 and October 2016 by setting from-month to 5 , from-year to 2015 and to-month and to-year to 10 and 2016 respectively.

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz
```

Or in python do:

```
import urllib.request
from urllib.parse import urlencode

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz"

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/search'
query_params = { 'from-year': 2015, 'from-month': 5, 'to-year':2015, ;'to-month': 10 }

# Encode query parameters
encoded_params = urlencode(query_params)

# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as resp
```


```
response_body = response.read()
print(response_body.decode())
```

## Combining Filters

To constrain results any of the above filters can be arbitrarily combined by appending them as HTTP query string parameters. The various filters supported correspond to the faceted search fields on the user interface. Filters are combined as a boolean `AND` operation. Additionally some filters support multiple values which corresponds to a boolean `OR` operation.

For example we can search on both `address` and `energy-band` by including both filters on the query string:

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy
```




## 4: Non-domestic Certificate API

This API can be called by making an authenticated HTTP `GET` request with an appropriate `Accept` header to the API endpoint:

```
https://epc.opendatacommunities.org/api/v1/non-domestic/certificate/:lmk-key
```

Where `:lmk-key` is the *LMK* key of a certificate from a search result or download. For example we can request the certificate with *LMK* key of `89931970062019053113040836250130` by making a request to the API at the URL:

```
https://epc.opendatacommunities.org/api/v1/non-domestic/certificate/8993197006201905311
```



**Note: This API recently changed, to expect an LMK key as the identifier instead of a certificate hash. [Read more](#).**

When a supported `Accept` header is set on the request it will return the specified certificate in the requested format. An example of how to request this certificate in CSV format using cURL is shown below:

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz
```

Or in python do:

```
import urllib.request
from urllib.parse import urlencode

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz"

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/certificate/8993197'
query_params = ''

# Encode query parameters
encoded_params = urlencode(query_params)

# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as resp:
    response_body = response.read()
    print(response_body.decode())
```

Descriptions of all the columns and fields found in the returned data can be found in the [glossary](#).

## 5: Non-domestic Recommendations API

Not all certificates contain recommendations for the property, but where they do they are available via this API route:


```
https://epc.opendatacommunities.org/api/v1/non-domestic/recommendations/:lmk-key
```



If the property has no recommendations a HTTP 404 not found status code will be returned. Note that this api route does not recognise the `application/zip` mime type.

Recommendations are keyed on the certificates `lmk-key` , which can be obtained from the certificate's `lmk-key` field.

```
$ curl -v -H "Accept: text/csv" -H "Authorization: Basic cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz
```



Or in python do:

```
import urllib.request
from urllib.parse import urlencode

token = "cmF5YW4uYXpoYXJpLjE3QHVjbC5hYy51azo3Yzg3ZjY0MGQ5ZDA0MWY0MjY4MTE0YWU1Yj1hY2Q3Nz"

headers = {
    'Accept': 'text/csv',
    'Authorization': f'Basic {token}'
}

# Define base URL and query parameters separately
base_url = 'https://epc.opendatacommunities.org/api/v1/non-domestic/recommendations/895'
query_params = ''

# Encode query parameters
encoded_params = urlencode(query_params)

# Append parameters to the base URL
full_url = f"{base_url}?{encoded_params}"

# Now make request
with urllib.request.urlopen(urllib.request.Request(full_url, headers=headers)) as response:
    response_body = response.read()
    print(response_body.decode())
```

