NLx Research Hub API Overview

The NLx Research Hub API provides access to job posting data provided by the [National Labor Exchange](http://nlxresearchhub.org) in partnership with DirectEmployers. The NLx has recently completed a migration of its historical data warehouse to a new data warehouse that improves data processing, reliability, and availability.

Data from the NLx Research Hub is provided through two APIs:

* **Synchronous API:** REST API for paging through data 50 records at a time.
* **Asynchronous API:** REST API for downloading large quantities of data from the NLx Research Hub (i.e. an entire month of jobs data from a single state or territory).

Below, we provide Python code examples for interacting with each API.

Before getting started, you will need to receive an API key. If you have not received an API key but expect to have received one, please reach out to [admin@nlxresearchhub.org](mailto:admin@nlxresearchhub.org).

This API key will provide access to either the Redacted or Un-Redacted data. Redacted data excludes the Occupation Code and any identifying information about the company posting each job opportunity. Un-Redacted data includes all elements available in the NLx Research Hub. Your access level will be provided to you along with your API key by email.

Once you have received an API key, you can interact with the APIs through the code examples below. This Overview document contains:

* Synchronous API code example in Python
* Asynchronous API code example in Python
* Asynchronous API code example in R
* Data Dictionary

## 

## **Synchronous API Code Example (Python)**

# Python package dependencies

import requests

import json

import pandas as pd

import time

api\_key = 'YOUR\_API\_KEY\_HERE'

params = {

'page': 1, # This allows you to page through results

'state': 'KY' #Use an upper-case two digit code for the state or territory you intend to query

}

headers = {

'X-API-Key': f'{api\_key}',

'Content-Type': 'application/json'

}

r = requests.get(f'https://api.nlxresearchhub.org/api/jobs/',headers=headers,params=params)

response\_json = r.json() # provides a JSON object in the form of a Python dictionary

## All of the jobs returned reside in a list inside the response 'data' key of the response JSON object

## Below is the first example Job JSON object (dict in python)

response\_json['data'][0]

## 

## **Asynchronous API Code Example (Python)**

# Python package dependencies

import requests

import json

import pandas as pd

import time

api\_key = 'YOUR\_API\_KEY\_HERE'

headers = {

    'X-API-Key': f'{api\_key}',

    'Content-Type': 'application/json'

}

# For the asynchronous API, we set our parameters and request the report

query\_params = {

    "state\_or\_territory":"OH", # Use an upper-case two digit code for the state or territory you intend to query

    "start":"2021-06-05", # Start date of the results (inclusive)

    "end":"2021-06-10", # End date of the results (inclusive). Please note that you cannot request more than 35 days of data at a time

# Dates are midnight to midnight, so 2021-06-01 to 2021-06-02 is one day of data, for 2021-06-01

    "format":"csv", # Currently supports csv and ndjson formats

"date\_column":"date\_compiled", # Or "date\_acquired"

}

# Requests the report from the Data Warehouse.

report\_request\_response = requests.post('https://api.nlxresearchhub.org/api/job\_reports/',

              headers=headers, data=json.dumps(query\_params))

# wait for the report to complete and query the API until the report is complete

# Sleep for 10 seconds between requests to avoid rate limitations

not\_done = True

while not\_done:

    print("Sleeping for 10 seconds to wait for report")

    time.sleep(10)

    report\_request\_json = report\_request\_response.json()

    report\_status\_url = report\_request\_json['data'][0]['url']

    report\_status\_response = requests.get(report\_status\_url,headers=headers)

    report\_status\_json = report\_status\_response.json()

    if report\_status\_json['data']:

        if report\_status\_json['data'][0]['status'] == 'done':

            not\_done = False

        else:

            print("Report Status: ", report\_status\_json['data'][0]['status'])

# output will contain the signed AWS S3 url that we download to a Pandas dataframe below

print("Report is done. Downloading...")

report\_output\_df = pd.read\_csv(report\_status\_json['data'][0]['resource']['link'])

print("Download completed.")

## # Code addendum if download does not automatically start: #

report\_output\_df.to\_csv('file\_name.csv')

# df.to\_csv('file\_name, encoding = 'utf-8-sig')

files.download('file\_name.csv')

## **Asynchronous API Code Example (R)**

library(httr)

library(jsonlite)

library(data.table)

library(lubridate)

library(tidyverse)

week\_ending\_date <- "2022-01-22"

state\_value <- "NV"

nlx\_api <- "Your-API-Key-Here"

# Set to only grab 1 week of postings at a time, can be manually changed.

nlx\_url <- "https://api.nlxresearchhub.org/api/job\_reports/"

week\_start\_date <- as.character(ymd(week\_ending\_date) - days(7))

# Creating NLX API download

# First, establish query parameters and POST initial request

# Second, enter loop to POST API request

# Third, check for reponse every 10 seconds

# Fourth, get resource download link when response status is "done"

# Finally, read results from CSV file into R

query\_params <- list(

state\_or\_territory = state\_value,

start = week\_start\_date,

end = week\_ending\_date,

format = 'csv'

)

nlx\_data <- POST(url = nlx\_url,

add\_headers('X-API-Key' = nlx\_api,

'Content-Type' = 'application/json'),

body = toJSON(query\_params, auto\_unbox = TRUE))

not\_done <- TRUE

while(not\_done) {

print("Waiting 10 seconds to check report...")

Sys.sleep(10)

nlx\_data <- POST(url = "https://api.nlxresearchhub.org/api/job\_reports/",

add\_headers('X-API-Key' = nlx\_api,

'Content-Type' = 'application/json'),

body = toJSON(query\_params, auto\_unbox = TRUE))

if(fromJSON(rawToChar(nlx\_data$content))$data[[3]] == "done"){

not\_done <- FALSE

}

}

download\_url <- fromJSON(rawToChar(nlx\_data$content))$data[[6]][1,1]

nlx\_import <- fread(download\_url)

## **Data Dictionary**

| **Field** | **Data Type** | **Definition** | **Sensitivity Classification** |
| --- | --- | --- | --- |
| system\_job\_id | bigserial | Numeric Job ID utilized by the Data Warehouse as a primary key | Unrestricted |
| job\_id | bigint | The job id assigned by DirectEmployers | Unrestricted |
| file\_name | varchar(255) | The name of the uploaded file from which this job was extracted | Unrestricted |
| file\_id | varchar(255) | Soure of the data ("JobCentral" is a legacy name for the NLx) | Unrestricted |
| date\_compiled | timestamp | The date and time the job was received from DirectEmployers in its most recent version (date file written).  NOTE: this is the recommended default for the parameter date\_column. | Unrestricted |
| date\_acquired | timestamp | The date DirectEmployers acquired the job record. | Unrestricted |
| created\_date | timestamp | Date this job was first created in NLx Data Warehouse or legacy database.  NOTE: records backfilled from 2020 may have a “created date” in October 2021. See the above recommendation to use date\_compiled for analysis. | Unrestricted |
| last\_updated\_date | timestamp | Date of most recent update from DirectEmployers data. | Unrestricted |
| source\_state | varchar(255) | The State from which DirectEmployers received the job - null for every job that's not a state job. | Unrestricted |
| expired | smallint | Indication that the job no longer appears in the data files, marked as expired after 3 consecutive days of absence from data files. | Unrestricted |
| expired\_date | timestamp | The date when the job stopped appearing in ongoing data files, marked after 3 consecutive days of absence from data files. | Unrestricted |
| fedcontractor | boolean | Federal Contractor status for company | Unrestricted |
| fein | varchar(255) | Federal Employer Identification Number - null if unknown | Restricted |
| title | varchar(255) | Job title of the job posting | Unrestricted |
| description | text | Job description of the job posting | Unrestricted |
| link | varchar(255) | Link-back “url” where the job can be found | Unrestricted |
| date\_acquired | timestamp | Date and time the job was first acquired by DirectEmployers | Unrestricted |
| zipcode | varchar(60) | Job posting zip code | Unrestricted |
| city | varchar(255) | Job posting city | Unrestricted |
| state | varchar(255) | Job posting state | Unrestricted |
| country | varchar(255) | Job posting country | Unrestricted |
| parameters\_positions\_max | integer | Number of positions available for that job posting - only available for state or posted jobs | Unrestricted |
| parameters\_duration\_max | varchar(255) | Schedule for this job posting (i.e. Part-Time, Full-Time, or Flexible) | Unrestricted |
| parameters\_hours\_per\_week\_max | integer | Number of hours per week expected for the job | Unrestricted |
| parameters\_shift\_max | varchar(255) | Day shift, night shift, or swing shift | Unrestricted |
| parameters\_salary\_unit | varchar(255) | Salary unit of the job posting | Unrestricted |
| parameters\_salary\_min | numeric(15,4) | Salary minimum of the job posting | Unrestricted |
| parameters\_salary\_max | numeric(15,4) | Salary maximum of the job posting | Unrestricted |
| classifications\_onet\_code | varchar(255) | O\*Net Occupation Code of the job posting | Restricted |
| classifications\_naics\_code | varchar(255) | NAICS Industry Code of the company posting the job (mostly unavailable) | Unrestricted |
| requirements\_min\_education | varchar(255) | Minimum education required for the job posting | Unrestricted |
| requirements\_experience | varchar(255) | Minimum experience required for the job posting | Unrestricted |
| requirements\_license | varchar(255) | License requirements for the job posting, if applicable | Unrestricted |
| requirements\_training | varchar(255) | Training requirements for the job posting, if applicable | Unrestricted |
| application\_company | varchar(255) | Company name | Restricted |
| application\_contact | varchar(255) | Contact at the company for the job posting | Unrestricted |
| application\_zipcode | varchar(255) | Zip code for the company | Unrestricted |
| application\_city | varchar(255) | City of the company | Unrestricted |
| application\_state | varchar(255) | State of the company | Unrestricted |
| application\_country | varchar(255) | Country of the company | Unrestricted |
| application\_address\_line\_1 | varchar(255) | Address line 1 of the company | Restricted |
| application\_address\_line\_2 | varchar(255) | Address line 2 of the company | Unrestricted |
| application\_address\_line\_3 | varchar(255), | Address line 3 of the company | Unrestricted |
| application\_address\_line\_4 | varchar(255), | Address line 4 of the company | Unrestricted |
| application\_method\_1 | varchar(255), | First application method allowed | Unrestricted |
| application\_method\_2 | varchar(255), | Second application method allowed, if applicable | Unrestricted |
| application\_method\_3 | varchar(255), | Third application method allowed, if applicable | Unrestricted |
| application\_method\_4 | varchar(255), | Fourth application method allowed, if applicable | Unrestricted |
| application\_method\_text\_1 | text, | Where the first application method can be accessed (i.e. URL for online applications or Fax Number for fax applications, etc.) | Unrestricted |
| application\_method\_text\_2 | text, | Where the second application method can be accessed (i.e. URL for online applications or Fax Number for fax applications, etc.) | Unrestricted |
| application\_method\_text\_3 | text, | Where the third application method can be accessed (i.e. URL for online applications or Fax Number for fax applications, etc.) | Unrestricted |
| application\_method\_text\_4 | text | Where the fourth application method can be accessed (i.e. URL for online applications or Fax Number for fax applications, etc.) | Unrestricted |