### **Download Ads from Meta Library APIד**

## **1. Getting Started**

### **1.1 Prerequisites**

Before accessing the API, ensure that you:

* Have a **Meta Developer Account**
* Have an **App registered** in the Meta Developer Portal
* Obtain an **Access Token** with the required permissions

### **1.2 Authentication**

The API requires OAuth authentication using a **Meta App Access Token**. You can generate a token using:

curl -X GET "https://graph.facebook.com/oauth/access\_token?client\_id={APP\_ID}&client\_secret={APP\_SECRET}&grant\_type=client\_credentials"

This token is required for all API requests.

## **2. API Endpoints & Usage**

### **2.1 Searching for Ads**

The primary endpoint to retrieve ads:

GET https://graph.facebook.com/{API\_VERSION}/ads\_archive

#### **Query Parameters**

* ad\_type – Filter by ad type (political, social issues, etc.)
* search\_terms – Keywords to find relevant ads
* ad\_reached\_countries – Filter by country code (e.g., US, UK)
* ad\_delivery\_date\_min/max – Filter by ad run dates
* limit – Set the number of results per request

#### **Example Request**

curl -X GET "https://graph.facebook.com/v19.0/ads\_archive?search\_terms=climate&ad\_reached\_countries=['US']&access\_token={ACCESS\_TOKEN}"

## **3. Data Response Structure**

The API response contains structured ad data, including:

{

"data": [

{

"id": "1234567890",

"ad\_creative\_body": "Support clean energy now!",

"ad\_creative\_link\_caption": "Join the movement",

"ad\_creative\_link\_title": "Go Green",

"ad\_creative\_link\_url": "https://example.com",

"page\_name": "Green Energy Campaign",

"ad\_reached\_countries": ["US"],

"spend": { "amount": 500, "currency": "USD" },

"impressions": { "lower\_bound": 10000, "upper\_bound": 50000 }

}

],

"paging": {

"next": "https://graph.facebook.com/v19.0/ads\_archive?after=XYZ"

}

}

## **4. Rate Limits & Best Practices**

* **Rate Limits**: Requests per hour depend on your app’s tier (check Meta's policies).
* **Pagination**: Use the "paging" parameter to navigate large datasets.
* **Error Handling**: Implement retries and handle common errors (400, 403, 500).

## **5. Additional Resources**

* **Meta Graph API Explorer**:<https://developers.facebook.com/tools/explorer>
* **Official API Docs**:<https://www.facebook.com/ads/library/api/>
* **Access & Compliance Policies**: Ensure compliance with Meta’s API policies for transparency and data usage.

Would you like a specific implementation guide in Python or another language?

Here is a **Python guide** on how to use the **Meta Library API** to search for ads.

## **1. Prerequisites**

Before using the API, ensure that you have:

* A **Meta Developer Account**
* A registered **App** in the Meta Developer Portal
* An **App Access Token**

## **2. Install Required Libraries**

Ensure you have requests installed:

pip install requests

## **3. Authentication**

To authenticate, obtain an **App Access Token**:

import requests

APP\_ID = "your\_app\_id"

APP\_SECRET = "your\_app\_secret"

def get\_access\_token():

url = f"https://graph.facebook.com/oauth/access\_token"

params = {

"client\_id": APP\_ID,

"client\_secret": APP\_SECRET,

"grant\_type": "client\_credentials"

}

response = requests.get(url, params=params)

return response.json().get("access\_token")

ACCESS\_TOKEN = get\_access\_token()

print("Access Token:", ACCESS\_TOKEN)

## **4. Fetch Ads from the Library**

To retrieve ads, use the /ads\_archive endpoint.

### **Example: Search for Ads**

def search\_ads(search\_term, country="US", limit=10):

url = "https://graph.facebook.com/v19.0/ads\_archive"

params = {

"search\_terms": search\_term,

"ad\_reached\_countries": [country],

"limit": limit,

"access\_token": ACCESS\_TOKEN

}

response = requests.get(url, params=params)

return response.json()

ads\_data = search\_ads("climate change")

print(ads\_data)

## **5. Parsing API Response**

The API returns JSON data. Here’s how to extract useful information:

def parse\_ads(data):

if "data" in data:

for ad in data["data"]:

print(f"Ad ID: {ad['id']}")

print(f"Page Name: {ad.get('page\_name', 'N/A')}")

print(f"Ad Text: {ad.get('ad\_creative\_body', 'N/A')}")

print(f"Link: {ad.get('ad\_creative\_link\_url', 'N/A')}")

print(f"Impressions: {ad.get('impressions', {}).get('lower\_bound', 'N/A')} - {ad.get('impressions', {}).get('upper\_bound', 'N/A')}")

print("-" \* 50)

else:

print("No ads found.")

parse\_ads(ads\_data)

## **6. Handling Pagination**

If there are more results, use pagination:

def fetch\_all\_ads(search\_term, country="US"):

url = "https://graph.facebook.com/v19.0/ads\_archive"

params = {

"search\_terms": search\_term,

"ad\_reached\_countries": [country],

"limit": 100,

"access\_token": ACCESS\_TOKEN

}

ads = []

while url:

response = requests.get(url, params=params).json()

ads.extend(response.get("data", []))

url = response.get("paging", {}).get("next", None) # Get next page URL

params = {} # No need to send params again with next page URL

return ads

all\_ads = fetch\_all\_ads("elections")

parse\_ads({"data": all\_ads})

## **7. Error Handling**

Handle API errors properly:

def safe\_request(url, params):

try:

response = requests.get(url, params=params)

response.raise\_for\_status()

return response.json()

except requests.exceptions.RequestException as e:

print(f"Error: {e}")

return None

## **8. Rate Limits & Best Practices**

* **Rate Limits**: Limit API requests per hour based on your app's tier.
* **Caching**: Store API responses to reduce requests.
* **Retry Mechanism**: Handle failures using exponential backoff.

### 

### **Saving Meta Library API Data to Google Sheets using Python**

To store data from the **Meta Library API** directly into **Google Sheets**, we will use **gspread**, a Python library for interacting with Google Sheets.

## **1. Install Required Libraries**

First, install the necessary dependencies:

pip install requests gspread google-auth google-auth-oauthlib google-auth-httplib2 pandas

## **2. Set Up Google Sheets API**

Follow these steps to enable API access to Google Sheets:

1. **Go to Google Cloud Console**: Google Cloud Console
2. **Create a new project** or select an existing one.
3. **Enable APIs**:
   * Enable **Google Sheets API**.
   * Enable **Google Drive API** (needed for authentication).
4. **Create a Service Account**:
   * Go to **IAM & Admin > Service Accounts**.
   * Click **Create Service Account**.
   * Grant the **Editor** role (for modifying sheets).
   * Generate and download the JSON key file.
5. **Share the Google Sheet**:
   * Open your Google Sheet.
   * Share it with the service account email (found in the JSON key file).

## **3. Authenticate and Connect to Google Sheets**

Use **gspread** to authenticate with the API:

import gspread

from google.oauth2.service\_account import Credentials

# Load credentials from JSON key file

SERVICE\_ACCOUNT\_FILE = "your\_service\_account.json" # Replace with your JSON file path

SCOPES = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]

credentials = Credentials.from\_service\_account\_file(SERVICE\_ACCOUNT\_FILE, scopes=SCOPES)

client = gspread.authorize(credentials)

# Open Google Sheet by name

SHEET\_NAME = "Meta Ads Data"

spreadsheet = client.open(SHEET\_NAME)

worksheet = spreadsheet.sheet1 # Select first sheet

## **4. Fetch Ads from Meta Library API**

Use the **Meta Ads Library API** to fetch data:

import requests

APP\_ID = "your\_app\_id"

APP\_SECRET = "your\_app\_secret"

def get\_access\_token():

url = f"https://graph.facebook.com/oauth/access\_token"

params = {

"client\_id": APP\_ID,

"client\_secret": APP\_SECRET,

"grant\_type": "client\_credentials"

}

response = requests.get(url, params=params)

return response.json().get("access\_token")

ACCESS\_TOKEN = get\_access\_token()

def search\_ads(search\_term, country="US", limit=10):

url = "https://graph.facebook.com/v19.0/ads\_archive"

params = {

"search\_terms": search\_term,

"ad\_reached\_countries": [country],

"limit": limit,

"access\_token": ACCESS\_TOKEN

}

response = requests.get(url, params=params)

return response.json()

ads\_data = search\_ads("climate change")

## **5. Save Data to Google Sheets**

Now, let’s extract the relevant data and insert it into Google Sheets:

import pandas as pd

def save\_to\_google\_sheets(data):

if "data" not in data:

print("No ads found.")

return

ads\_list = []

for ad in data["data"]:

ads\_list.append([

ad.get("id", ""),

ad.get("page\_name", "N/A"),

ad.get("ad\_creative\_body", "N/A"),

ad.get("ad\_creative\_link\_url", "N/A"),

ad.get("spend", {}).get("amount", "N/A"),

ad.get("impressions", {}).get("lower\_bound", "N/A"),

ad.get("impressions", {}).get("upper\_bound", "N/A")

])

# Convert to DataFrame

df = pd.DataFrame(ads\_list, columns=["Ad ID", "Page Name", "Ad Text", "Link", "Spend", "Impressions Min", "Impressions Max"])

# Update Google Sheets

worksheet.clear() # Clear existing data

worksheet.update([df.columns.values.tolist()] + df.values.tolist()) # Insert new data

print("Data successfully written to Google Sheets!")

# Save ads data to Google Sheets

save\_to\_google\_sheets(ads\_data)

## **6. Automating Data Updates**

To automate the process:

* Run the script on a **cron job (Linux/macOS)** or **Task Scheduler (Windows)**.
* Use **Google Cloud Functions** to trigger updates periodically.

### **Final Thoughts**

This solution allows you to: ✅ **Fetch ads** from the Meta Library API  
 ✅ **Extract key ad information** ✅ **Store data directly into Google Sheets**

Would you like a version that runs on a schedule or additional features like filtering by date range?

### **Automating Daily Data Fetching & Storing Multiple Pages/Websites in Google Sheets**

### To enhance the script, we will: ✅ **Schedule it to run daily** using a cron job or Windows Task Scheduler. ✅ **Allow multiple pages/websites** for tracking. ✅ **Append new data instead of overwriting**.

## **1. Modify the Script to Support Multiple Pages/Websites**

### Update the function to accept a list of **page names** or **websites** as input.

### import gspread

### import requests

### import pandas as pd

### from google.oauth2.service\_account import Credentials

### from datetime import datetime

### 

### # Load Google Sheets Credentials

### SERVICE\_ACCOUNT\_FILE = "your\_service\_account.json" # Replace with your JSON file path

### SCOPES = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]

### 

### credentials = Credentials.from\_service\_account\_file(SERVICE\_ACCOUNT\_FILE, scopes=SCOPES)

### client = gspread.authorize(credentials)

### 

### # Open Google Sheet

### SHEET\_NAME = "Meta Ads Data"

### spreadsheet = client.open(SHEET\_NAME)

### worksheet = spreadsheet.sheet1 # Select first sheet

### 

### # Meta API Credentials

### APP\_ID = "your\_app\_id"

### APP\_SECRET = "your\_app\_secret"

### 

### def get\_access\_token():

### url = f"https://graph.facebook.com/oauth/access\_token"

### params = {

### "client\_id": APP\_ID,

### "client\_secret": APP\_SECRET,

### "grant\_type": "client\_credentials"

### }

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

### return response.json().get("access\_token")

### 

### ACCESS\_TOKEN = get\_access\_token()

### 

### def fetch\_ads\_by\_pages(pages, country="US", limit=10):

### """Fetch ads from Meta Library API for multiple pages or websites."""

### all\_ads = []

### 

### for page in pages:

### print(f"Fetching ads for: {page}")

### url = "https://graph.facebook.com/v19.0/ads\_archive"

### params = {

### "search\_terms": page,

### "ad\_reached\_countries": [country],

### "limit": limit,

### "access\_token": ACCESS\_TOKEN

### }

### response = requests.get(url, params=params).json()

### 

### if "data" in response:

### for ad in response["data"]:

### all\_ads.append([

### ad.get("id", ""),

### page, # Store the page/website being tracked

### ad.get("page\_name", "N/A"),

### ad.get("ad\_creative\_body", "N/A"),

### ad.get("ad\_creative\_link\_url", "N/A"),

### ad.get("spend", {}).get("amount", "N/A"),

### ad.get("impressions", {}).get("lower\_bound", "N/A"),

### ad.get("impressions", {}).get("upper\_bound", "N/A"),

### datetime.now().strftime("%Y-%m-%d %H:%M:%S") # Timestamp

### ])

### 

### return all\_ads

### 

### def save\_to\_google\_sheets(data):

### """Save new ad data to Google Sheets, appending new entries instead of overwriting."""

### if not data:

### print("No ads found.")

### return

### 

### df = pd.DataFrame(data, columns=["Ad ID", "Search Query", "Page Name", "Ad Text", "Link", "Spend", "Impressions Min", "Impressions Max", "Timestamp"])

### 

### # Append data to the existing sheet (without deleting old data)

### worksheet.append\_rows(df.values.tolist())

### print(f"✅ {len(data)} ads successfully added to Google Sheets!")

### 

### # Define pages or websites to track

### PAGES\_TO\_TRACK = ["Tesla", "Nike", "climate change", "example.com"]

### 

### # Fetch and save data

### ads\_data = fetch\_ads\_by\_pages(PAGES\_TO\_TRACK, country="US", limit=20)

### save\_to\_google\_sheets(ads\_data)

## **2. Automate the Script to Run Daily**

### **Linux/macOS (Using Cron Job)**

### Open the terminal and type: crontab -e

### Add the following line to run the script daily at 12:00 AM: 0 0 \* \* \* /usr/bin/python3 /path/to/your\_script.py

### Replace /path/to/your\_script.py with the actual path of your script.

### **Windows (Using Task Scheduler)**

### Open **Task Scheduler**.

### Click **Create Basic Task**.

### Set the **Trigger** to **Daily**.

### Set the **Action** to **Start a Program** and select **python.exe**.

### In the **Add arguments** field, enter the script’s path: "C:\path\to\your\_script.py"

### Click **Finish**.

## **3. Features & Enhancements**

### **Multiple Pages/Websites**: You can add or remove names in the PAGES\_TO\_TRACK list.

### **Daily Updates**: New data is appended, allowing historical tracking.

### **Timestamped Data**: Each entry is saved with a date/time for trend analysis.

### Would you like additional features like **email alerts** when new ads are detected? 🚀

### 

### **Extracting Ads from Pages vs. Websites Using the Meta Library API**

The **Meta Ads Library API** allows you to retrieve ads based on different criteria, such as **Facebook Pages** or **websites (domains)**. Understanding the difference between the two approaches helps in choosing the right method based on your use case.

## **1. Extracting Ads from a Facebook Page**

If you want to track ads run by a **specific Facebook Page**, you need to use the **page ID** or **page name** in your search.

### **How It Works**

* The API returns **ads published by that specific page**.
* You can search by the **page name** (e.g., "Nike" or "Tesla").
* The ads will include data such as the **text, images, impressions, spend, and the URL linked to the ad**.

### **Example Request (Using Page Name)**

import requests

ACCESS\_TOKEN = "your\_access\_token"

PAGE\_NAME = "Tesla"

url = "https://graph.facebook.com/v19.0/ads\_archive"

params = {

"search\_terms": PAGE\_NAME,

"ad\_reached\_countries": ["US"],

"limit": 10,

"access\_token": ACCESS\_TOKEN

}

response = requests.get(url, params=params)

ads\_data = response.json()

print(ads\_data)

### **When to Use This Approach**

✅ You want to **monitor a competitor's ads** on Facebook.  
 ✅ You want to **analyze an advertiser’s ad spending trends**.  
 ✅ You are tracking **a political figure, campaign, or organization**.

## **2. Extracting Ads from a Website (Domain-Based Search)**

If you want to track ads that **lead to a specific website**, you can use the **domain name** (e.g., "example.com") as the search term.

### **How It Works**

* The API will return **ads containing links to the specified website**.
* This includes **ads from different advertisers** that promote the website.
* Useful for tracking **affiliates, brand mentions, and product promotions**.

### **Example Request (Using Website URL)**

DOMAIN = "example.com"

url = "https://graph.facebook.com/v19.0/ads\_archive"

params = {

"search\_terms": DOMAIN,

"ad\_reached\_countries": ["US"],

"limit": 10,

"access\_token": ACCESS\_TOKEN

}

response = requests.get(url, params=params)

ads\_data = response.json()

print(ads\_data)

### **When to Use This Approach**

✅ You want to **see all advertisers promoting a particular website**.  
 ✅ You are tracking **affiliate marketing or reseller activity**.  
 ✅ You need to **analyze brand mentions** across various campaigns.

## **3. Key Differences Between Page-Based and Website-Based Search**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Page-Based Search** | **Website-Based Search** |
| **Ads from a specific advertiser** | ✅ Yes | ❌ No |
| **Ads containing a website URL** | ❌ No | ✅ Yes |
| **Competitor tracking** | ✅ Best for monitoring brands | ✅ Useful for tracking links to a competitor’s site |
| **Affiliate monitoring** | ❌ Not applicable | ✅ Shows multiple advertisers linking to a domain |
| **Political ad transparency** | ✅ Shows ads from politicians or campaigns | ❌ Not relevant |

### **Conclusion**

* Use **Page-Based Search** if you need to track a **specific advertiser** or **competitor’s ads**.
* Use **Website-Based Search** if you want to **see all ads that direct traffic to a specific domain**, regardless of the advertiser.

Would you like a script that can **fetch both types of ads and save them into Google Sheets** automatically? 🚀

Here's a **Python script** that fetches **both Page-Based and Website-Based ads** from the Meta Ads Library API and saves them into **Google Sheets automatically**. This script allows you to track multiple **Facebook Pages** and **Websites (Domains)** daily.

## **1. Install Required Libraries**

Ensure you have the necessary dependencies installed:

pip install requests gspread google-auth google-auth-oauthlib google-auth-httplib2 pandas

## **2. Set Up Google Sheets API**

Follow these steps:

1. **Go to Google Cloud Console**: Google Cloud Console
2. **Enable APIs**:
   * Enable **Google Sheets API**.
   * Enable **Google Drive API**.
3. **Create a Service Account** and **download the JSON key file**.
4. **Share the Google Sheet** with the service account email (found in the JSON file).

## **3. Python Script: Fetch Ads and Save to Google Sheets**

This script:

* Fetches ads from both **Facebook Pages** and **Websites**.
* Stores results in **Google Sheets**.
* Appends new data instead of overwriting.

import gspread

import requests

import pandas as pd

from google.oauth2.service\_account import Credentials

from datetime import datetime

# Load Google Sheets Credentials

SERVICE\_ACCOUNT\_FILE = "your\_service\_account.json" # Replace with your JSON file path

SCOPES = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]

credentials = Credentials.from\_service\_account\_file(SERVICE\_ACCOUNT\_FILE, scopes=SCOPES)

client = gspread.authorize(credentials)

# Open Google Sheet

SHEET\_NAME = "Meta Ads Data"

spreadsheet = client.open(SHEET\_NAME)

worksheet = spreadsheet.sheet1 # Select first sheet

# Meta API Credentials

APP\_ID = "your\_app\_id"

APP\_SECRET = "your\_app\_secret"

def get\_access\_token():

"""Retrieve Meta API access token"""

url = f"https://graph.facebook.com/oauth/access\_token"

params = {

"client\_id": APP\_ID,

"client\_secret": APP\_SECRET,

"grant\_type": "client\_credentials"

}

response = requests.get(url, params=params)

return response.json().get("access\_token")

ACCESS\_TOKEN = get\_access\_token()

def fetch\_ads(search\_terms, search\_type="page", country="US", limit=10):

"""Fetch ads by Page Name or Website"""

all\_ads = []

for term in search\_terms:

print(f"Fetching ads for: {term} ({search\_type})")

url = "https://graph.facebook.com/v19.0/ads\_archive"

params = {

"search\_terms": term,

"ad\_reached\_countries": [country],

"limit": limit,

"access\_token": ACCESS\_TOKEN

}

response = requests.get(url, params=params).json()

if "data" in response:

for ad in response["data"]:

all\_ads.append([

ad.get("id", ""),

search\_type, # "page" or "website"

term, # The tracked Page or Website

ad.get("page\_name", "N/A"),

ad.get("ad\_creative\_body", "N/A"),

ad.get("ad\_creative\_link\_url", "N/A"),

ad.get("spend", {}).get("amount", "N/A"),

ad.get("impressions", {}).get("lower\_bound", "N/A"),

ad.get("impressions", {}).get("upper\_bound", "N/A"),

datetime.now().strftime("%Y-%m-%d %H:%M:%S") # Timestamp

])

return all\_ads

def save\_to\_google\_sheets(data):

"""Save new ad data to Google Sheets, appending new entries instead of overwriting."""

if not data:

print("No ads found.")

return

df = pd.DataFrame(data, columns=["Ad ID", "Search Type", "Search Query", "Page Name", "Ad Text", "Link", "Spend", "Impressions Min", "Impressions Max", "Timestamp"])

# Append data to the existing sheet

worksheet.append\_rows(df.values.tolist())

print(f"✅ {len(data)} ads successfully added to Google Sheets!")

# Define Pages and Websites to track

PAGES\_TO\_TRACK = ["Tesla", "Nike", "Apple"]

WEBSITES\_TO\_TRACK = ["tesla.com", "nike.com", "apple.com"]

# Fetch and save data for Pages

ads\_from\_pages = fetch\_ads(PAGES\_TO\_TRACK, search\_type="page", country="US", limit=20)

save\_to\_google\_sheets(ads\_from\_pages)

# Fetch and save data for Websites

ads\_from\_websites = fetch\_ads(WEBSITES\_TO\_TRACK, search\_type="website", country="US", limit=20)

save\_to\_google\_sheets(ads\_from\_websites)

## **4. Automate Daily Execution**

### **Linux/macOS (Using Cron Job)**

Open the terminal and type:  
 crontab -e

Add the following line to run the script daily at 12:00 AM:  
 0 0 \* \* \* /usr/bin/python3 /path/to/your\_script.py

1. Replace /path/to/your\_script.py with your script’s actual location.

### **Windows (Using Task Scheduler)**

1. Open **Task Scheduler**.
2. Click **Create Basic Task**.
3. Set the **Trigger** to **Daily**.
4. Set the **Action** to **Start a Program** and select **python.exe**.

In the **Add arguments** field, enter the script’s path:  
 "C:\path\to\your\_script.py"

1. Click **Finish**.

## **5. Features & Enhancements**

✅ **Tracks both Pages & Websites** – See ads from specific brands and URLs.  
 ✅ **Appends data instead of overwriting** – Keeps a historical record.  
 ✅ **Runs daily automatically** – Keeps your ad data updated.

Would you like an **email notification** when new ads are detected? 🚀