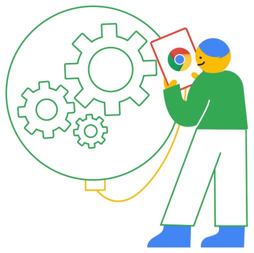
chrome enterprise

How to use Chrome Browser Cloud Management's Takeout API Service Script



Contents

chrome enterprise

How to setup the Admin SDK

1. Enable the Admin SDK API (if not enabled) in the Google Developer Console by following this [link] and selecting the project you wish to enable the API or create a new one.
    - Note that your Google admin account might require additional rights for API access, More information about API rights

2. Open the Service accounts page. If prompted, **select a project.**

3. Click + **Create Service Account**, enter a name and description for the service account. You can use the default service account ID, or choose a different, unique one. When done click **Create.**

4. In the Service account permissions make sure to **grant the service account the "Service Account User" role.**

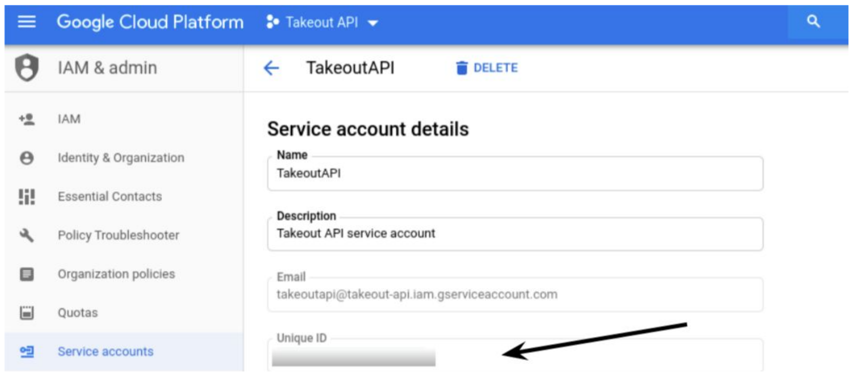
5. On the Grant users access to this service account screen, scroll down to the **Create key** section. Click **+ Create key.**

6. In the side panel that appears, select the format for your key: **JSON** is recommended.

7. Click **Create.** Your new public/private key pair is generated and downloaded to your machine; it serves as the only copy of this key.

8. Click **Close** on the **Private key saved to your computer** dialog, then click **Done** to return to the table of your service accounts.

9. Go to **service accounts** and copy the **Unique ID.**



Service account details

Name
TakeoutAPI

Description
Takeout API service account

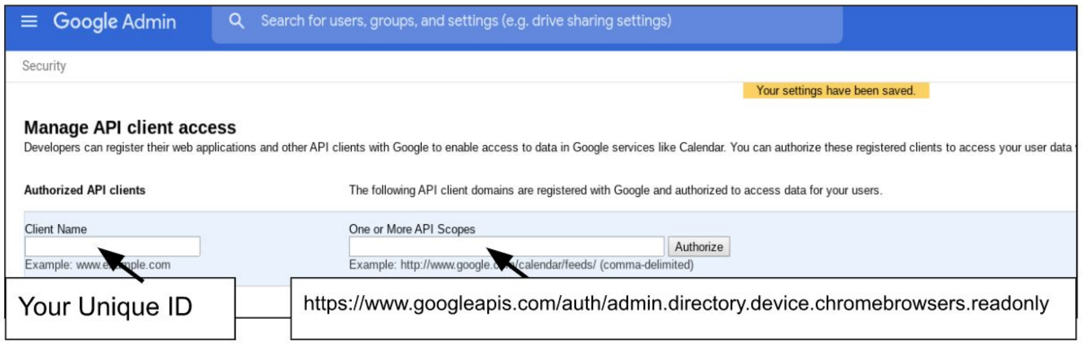
Email
takeoutapi@takeout-api.iam.gserviceaccount.com

Unique ID
[redacted]

Adding the Scopes to your Google Admin Console

1. Open the google admin console where you want to enable the API under Security > Advanced settings > Manage API client access.
    a. On this page, the Client name corresponds to the Unique ID of your service account (gathered from step 10).
    b. Enter in the following for One or more API Scopes and hit authorize: https://www.googleapis.com/auth/admin.directory.device.chromebrowsers.readonly

Note: The Google API library fully supports and tests on Python 3.4, 3.5, 3.6 and 3.7. This library may work on later versions of 3, but we do not currently run tests against those versions.


Google Admin
Search for users, groups, and settings (e.g. drive sharing settings)
Security
Your settings have been saved.

Manage API client access

Developers can register their web applications and other API clients with Google to enable access to data in Google services like Calendar. You can authorize these registered clients to access your user data

Authorized API clients

The following API client domains are registered with Google and authorized to access data for your users.

Client Name

Example: www.example.com

One or More API Scopes

Authorize

Example: http://www.google.com/calendar/feeds/ (comma-delimited)

Your Unique ID

https://www.googleapis.com/auth/admin.directory.device.chromebrowsers.readonly

Running the Script on Chrome OS/Linux

Note: you can ignore Step 1 if running on Linux.
Python comes installed by default on the Chrome OS version of Linux.

1. Open the Linux settings via the Chrome OS and activate the Linux Beta and install it. More information is located here - Might take around ~1gb of space or more on your machine once Python is installed.

2. Type : sudo apt install python3-pip

3. pip3 install google-api-python-client (more instructions here)

4. Download the Service Account Key from your Google Developer Console and save it locally within your Linux folder.

5. Browse to the takeout script page, copy the text and save it in your linux directory and rename the extension as .py
   - Linux users can download the script directly via this cmd
   - curl: https://cs.chromium.org/codesearch/f/chromium/src/docs/enterprise/extension_query.py > extension_query.py

6. Run the following command replacing the sections in blue with the name of your key file and email address of your admin account that you enabled the admin SDK with.
   - python3 extension_query.py --service_account_key_path <yourkeyfilename>.json --admin_email <adminaccountname>@<yourdomain.com>

7. The output should be saved as extension_list.csv that you can open in Excel or Google Sheets.

Running the Script on Windows

1. Download the latest version of Python from their official site (if you don't already have it installed)

2. Run the installer (run as administrator)
   - make sure that you select custom installation and select add python to environment variables if you want to run this via cmd line and make sure that install pip was selected.

3. Open an administrator Cmd to verify that it was correctly installed by typing the cmd:
   - Python
   - make sure that that you exit() to leave the python shell

4. Verify that PIP was correctly installed by typing the cmd : pip -v

5. Install the google api library via typing the cmd:
   - pip install google-api-python-client
   - If you get a syntax error you need to leave the python shell via exit() and then retype in the normal cmd line interface.

6. Browse to the takeout script page, copy the text and save it in your specified directory and rename the extension as .py
   - Download and place your keyfile in the same file location saved as a .json file.

7. Run the following command replacing the sections in blue with:
   1. The path to your key file and the .py script
   2. The name of your key file
   3. Email address of your admin account that you enabled the admin SDK with.
   - Python.exe <path to script>\extension_query.py --service_account_key_path <yourkeyfilename>.json --admin_email <adminaccountname>@<yourdomain.com>

8. The output should be saved as extension_list.csv that you can open in Excel or Google Sheets.

Running the Script on macOS

1. MacOS comes with Python pre installed but it can be out of date. We will need to update it:
    - Go to Python.org downloads page and download the latest Python installer package
    - Run the Python installer package and install Python 3 onto the Mac

2. Open a Terminal window

3. Run the following command to install pip3 on your Mac:
    - pip3
    - You will get prompted to install the command line developer tools. Click Install at the prompt and then Agree at the next pop up
    - MacOS will automatically download and install the required software.
    - This may take a few minutes. When install is complete, click Done at the prompt

4. Update pip3 to the latest version:
    - sudo pip3 install --upgrade pip

5. Run the command to install the python client for pip3:
    - sudo pip3 install google-api-python-client (more instructions here)

6. Download the Service Account Key from your Google Developer Console and save it locally within your macOS folder.

7. Browse to the takeout script page, copy the text and save it in your macOS directory and rename the extension as .py. Make sure the file is in the same directory as your Service Account Key.
    - macOS users can download the script directly via this Terminal cmd
    - curl https://cs.chromium.org/codesearch/f/chromium/src/docs/enterprise/extension_query.py > extension_query.py

8. Change the directory in Terminal to the download folder:
    - cd Downloads (if using a different folder, just navigate to the correct directory using the cd command)

9. Run the following command replacing the sections in blue with the name of your key file and email address of your admin account. This can be any Admin account in the domain that has the required privileges to view the browsers.
   - python3 extension_query.py --service_account_key_path <yourkeyfilename>.json --admin_email <adminaccountname>@<yourdomain.com>

10. The output should be saved as extension_list.csv that you can open in Excel or Google Sheets.
    - Note: failure to change the directory in Terminal at step 8 will cause the file to be save in the root folder of your user profile

Reviewing the data

Here is a overview of the fields that are available in the output:

id	Name	num_installed	num_disabled	num_forced
The Unique ID of the extension	The Name of the extension	The amount of times the extensions has been installed	Count of # of times the extension has been disabled by an admin	Count of # of times the extension has been Force installed by an admin
permissions	**installed**	**disabled**	**forced**	**num_permissions**
The rights that the extension requires to run as declared in the manifest	Which machines have the extension installed	Which machines have the extension disabled	Which machines have the extension forced	How many permissions does the extension require to run

Additional Command Line Arguments for the Script

The following additional command line arguments can be used to control the behavior of the script (these apply to all platforms):

- **extension_list_csv:** You can can specify the exact output file location of the finished extension analysis
- **max_browsers_to_process:** This option will limit the number of browsers details to fetch before the analysis calculations are done. Since each request to the server typically returns 100 results, the calculated result may return up to 100 more browsers than the specified argument.