

Set up your Merchant Center Account

The Content API for Shopping allows apps to interact directly with the [Merchant Center platform](#), vastly increasing the efficiency of managing large or complex Merchant Center accounts. Some typical use cases include:

- Automated account management
- Product management on a per-product basis
- Datafeed scheduling
- Managing complex tax and shipping settings

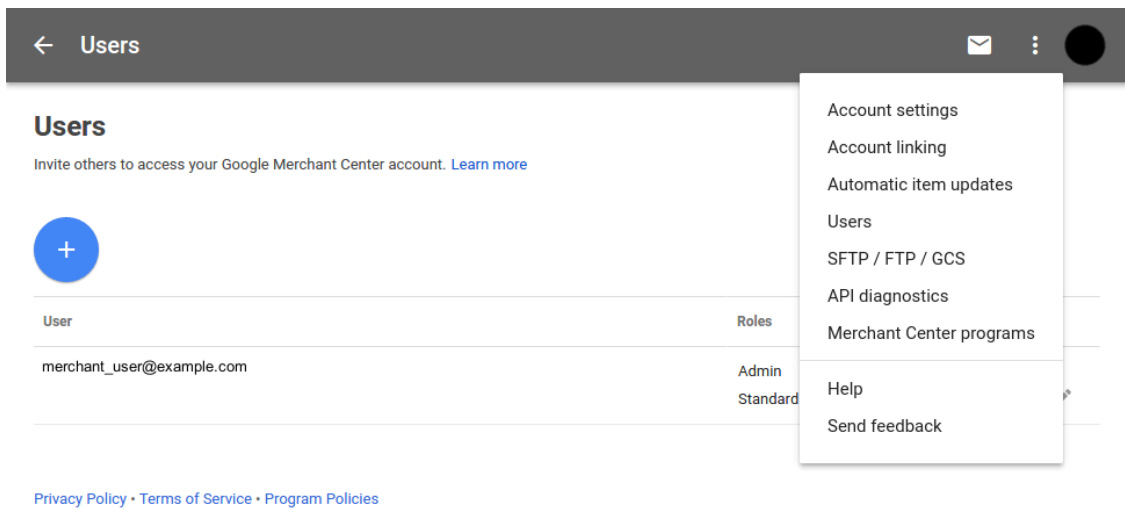
With the Content API for Shopping you can build software that manages accounts from the customer level down to the product level. The API can do almost everything the Merchant Center website does, but *programmatically*. To help you get started, we offer [code samples](#) in Java, .NET, Python, PHP, Ruby, and Go.

This guide will help you create your first application using the Content API. Before you can start coding, there are a few things you need to do.

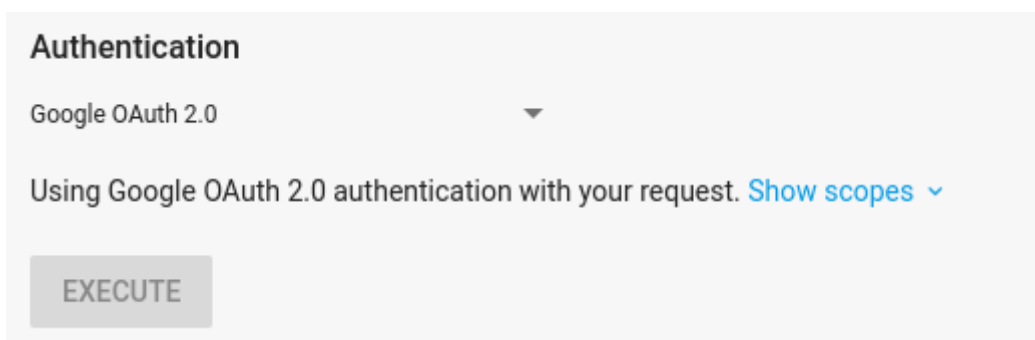
1. Create a [Merchant Center Account](#).
2. [Verify and claim your website URL](#).
3. Configure your [Tax & Shipping settings](#).
4. Be sure to remember which email you used to create this merchant center account, as you will need that later. In the screenshots, you'll see it represented as `merchant_user@example.com`.

Make your first API request (without a line of code!)

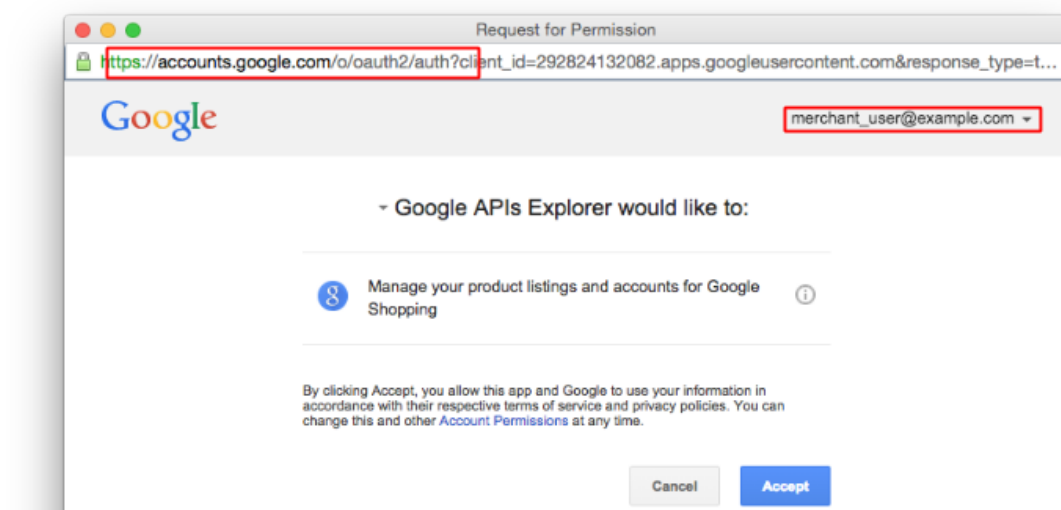
1. Head over to the [API Explorer](#) in our developer documentation for `Products.list`.
2. Ensure that the logged in Google account is listed in the 'Users' setting, available from the overflow menu.



3. In the API Explorer, make sure the Authentication setting says **Google OAuth 2.0**:




4. Click **Accept** to authorize a Google-created API Console Project to temporarily access your Merchant Center account on your behalf to make an API request. Note that this screen should come from google.com, and you should be signed in to your Merchant Center email address.



5. Enter your Merchant ID from your Merchant Center account into the 'merchantId' field of the form, and click **Execute**.

Try it!

Use the APIs Explorer below to call this method on live data and see the response.

Authorize requests using OAuth 2.0: 

merchantId The ID of the managing account. (string)

maxResults The maximum number of products to return in the response, used for paging. (integer)

pageToken The token returned by the previous request. (string)

fields Selector specifying which fields to include in a partial response. [Use fields editor](#)

bold red = required

content.products.list executed one minute ago time to execute: 574 ms

Request

GET https://www.googleapis.com/content/v2/1234567890/products?key={YOUR_API_KEY}

Response

```
200 OK
- SHOW HEADERS -
{
  "kind": "content#productsListResponse"
}
```

6. You should see a successful response. Congratulations! You've made your first Shopping API request.
7. If you have no products in your Merchant Center yet, there won't be any in the list. But never fear, we'll add some soon. At this point, you can hop to other requests and take a look at what they do. Since this is a RESTful API, all requests work similarly. Good next steps include [Products.insert](#), [Inventory.set](#), and [Accountstatuses.get](#).

Create a Google API Console Project

Requests to the Content API for Shopping are made through your API Console project. Here we assume you will be accessing your own Merchant Center account, so we suggest using [service accounts](#) to simplify the authentication flow. Please check out the [Service Accounts](#) Guide for details on how to set up a new service account to use with your Merchant Center account.

Note: If you are interested in making calls on behalf of clients with their own Merchant Center accounts, then please see the [Authorize Requests](#)

guide. However, for trying out the Content API, we suggest setting up a [testing account](#) and using that until you have finished development.

Your first authorized API request using Python

Now that you've set up a service account and added the service account ID to your Merchant Center account, you can access your account using the available [Samples code](#). This guide will use the Python samples to demonstrate using the API.

1. Download our [Python samples](#) from our GitHub page.
2. From within the `python` directory, use `pip` to install needed package dependencies. More detailed instructions are available in the accompanying `README.md` files.

```
$ pip install -r requirements.txt
```

3. Run the `shopping/content/products/list.py` file, similar to what we did in the API Sandbox earlier.

```
$ python -m shopping.content.products.list
```

4. If you haven't followed the configuration directions in the `README` accompanying the source files, you will get an error that no sample configuration could be found. Follow the directions in the `README` to create the configuration directory and to populate it with the appropriate information. The JSON file you downloaded while setting up your service account should be placed in that directory with the filename `service-account.json`.

5. Now that you have set up the samples configuration, run `shopping/content/products/list.py` again.

```
$ python -m shopping.content.products.list
```

6. At this point, the Python script should finish executing, and print out either a list of products that it found, or a message that there were no products in your account. If you get an error such as "User cannot access account 1234567890", it's most likely because you have not set up the service account user as an authorized user in Merchant Center. From the API's standpoint, the API Console project sent a request on behalf of a user who was not on the list of authorized Merchant Center administrators, so it would, of course, reject that request.

7. You can experiment with running `shopping/content/products/list.py` as many times as you want, since it is a read-only request. You can also

explore the many other samples that we've created for both Python and other languages.

At this point, you've got a well suited environment for developing with the Content API for Shopping. You can now move on to learning more about [Making Requests](#) and [Best Practices](#).

Viewed using [Just Read](#)