# **Python SDK: Overview**

## **Introduction**

The Python SDK makes it incredibly simple to interact with all of OrderCloud’s features. Think of it as a convenient wrapper around all of the API’s endpoints. All the methods for the most part are a 1:1 reflection of the API. So if you wanted to get the currently authenticated user for example (which happens to be a GET request at v1/me ) it would be:

OrderCloud.MeApi.get();

This guide will cover how to install and get started with the Python SDK, as well as some helpful tips when working with it!

### **Installation**

The easiest way to install the SDK is to use the Python package manager, pip.

pip install OrderCloud

Alternatively you can clone the [Git repository](https://github.com/ordercloud-api/python-sdk) and copy the *OrderCloud/* folder in your project directory.

Once installed, just import the package:

import OrderCloud

And you’re good to go!

### **Configuration**

Before you can actually make any API calls, you need to configure some minimal information to communicate with a specific Organization or Buyer company.

All configuration is stored inside a configuration object. So you can use it like this:

import OrderCloud  
OrderCloud.configuration.client\_id = “Your\_Client\_ID”  
OrderCloud.configuration.scopes = [“FullAccess”,”DevCenterImpersonate”]

client\_id defines the API Client and can be retrieved from any organization or buyer API Client list in the dashboard.

scopes is an array of claims the app will request from the Oauth server. Claims are unique for each API Client and can be limited further via each user's Security Profile.

Now you can authenticate and make an API call:

OrderCloud.auth.login("USERNAME","PASSWORD")  
print(OrderCloud.MeApi.get())#Should print your user information

## **Important Nuances**

There are some additional things to know about how the Python SDK interfaces with the API. To make API calls a little less tedious, some things are abstracted away. This is important to keep in mind.

### **Authentication**

The easiest way to authenticate is through the OrderCloud.auth.login function. This takes a username and a password, make a request to grab an access token and stores it in OrderCloud.configuration.access\_token. If you’re not sure what an "access token" is or why you need one, please read our guide on [Authentication Workflows](https://devcenter.ordercloud.io/).

If you ever need to set the access token manually (such as if you were saving the token in a cookie and want to load it) you can always just directly set it on OrderCloud.configuration.access\_token. As long as there is a valid access token there, you can skip the authentication step and make all your API calls. The token gets added automatically as a header to each request.

The SDK also supports other authentication workflows. The OrderCloud.auth.login function takes a third parameter, client\_secret for when you have that set in your app’s setting in the developer center.

You can also authenticate using just the client\_id and client\_secret with the OrderCloud.auth.authenticate function. A full example would look like this:

import OrderCloud  
OrderCloud.configuration.client\_id = “Your\_Client\_ID”  
OrderCloud.configuration.scopes = [“FullAccess”,”DevCenterImpersonate”]  
OrderCloud.auth.authenticate(“Client Secret ID”)  
print(OrderCloud.MeApi.get())  
  
Alternatively if your app does not have a client\_secret , you can call OrderCloud.auth.authenticate() with no arguments and get back an access token. This would be the workflow for when you have a public store that anyone can access.

### **Impersonation**

The SDK also provides a class to make impersonation a little more convenient through the OrderCloud.impersonation class. This class has two functions:

OrderCloud.impersonation.start(access\_token)

Now any API calls you make will be on behalf of that user. To stop the impersonation, just do:

OrderCloud.impersonation.stop()

Even though it was stopped, it still saves the original access token. So the next time you call start, you can call it with no arguments and it will use the last one.

To acquire an access token to use when impersonating, you can use the [UserApi.get\_access\_token](https://github.com/ordercloud-api/python-sdk/blob/master/docs/UserApi.md#get_access_token) method. Here is a full working example:

#Import the library  
import OrderCloud as oc  
#Configure client id and scopes  
oc.configuration.client\_id = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
oc.configuration.scopes = ["FullAccess"]  
#Acquire and store the access token  
oc.auth.login("user","pass")  
print(oc.MeApi.get())#Should print out the user you just logged in with  
#Get an impersonation token   
token\_request = oc.ImpersonateTokenRequest(client\_id="[INSERT YOUR BUYER APP CLIENT ID]",claims=[])  
resp = oc.UserApi.get\_access\_token("buyer\_id","user\_id",token\_request)  
print(resp.access\_token) #This is our access token  
#Start the impersonation  
oc.impersonation.start(resp.access\_token)  
print(oc.MeApi.get()) #Should give us the information of the user we just impersonated  
oc.impersonation.stop()  
print(oc.MeApi.get()) #Our original user is back now

## **Conclusion**

Hopefully this document has provided you with enough information to get you started! Remember that the [API documentation](https://devcenter.ordercloud.io/docs) should be your go-to guide for working with the API. The [Python SDK repository](https://github.com/ordercloud-api/python-sdk) also has documentation and code examples specific to Python.

If you notice anything wrong or have any questions, don’t hesitate to reach out to us, and happy coding!