April 2024

API Guide

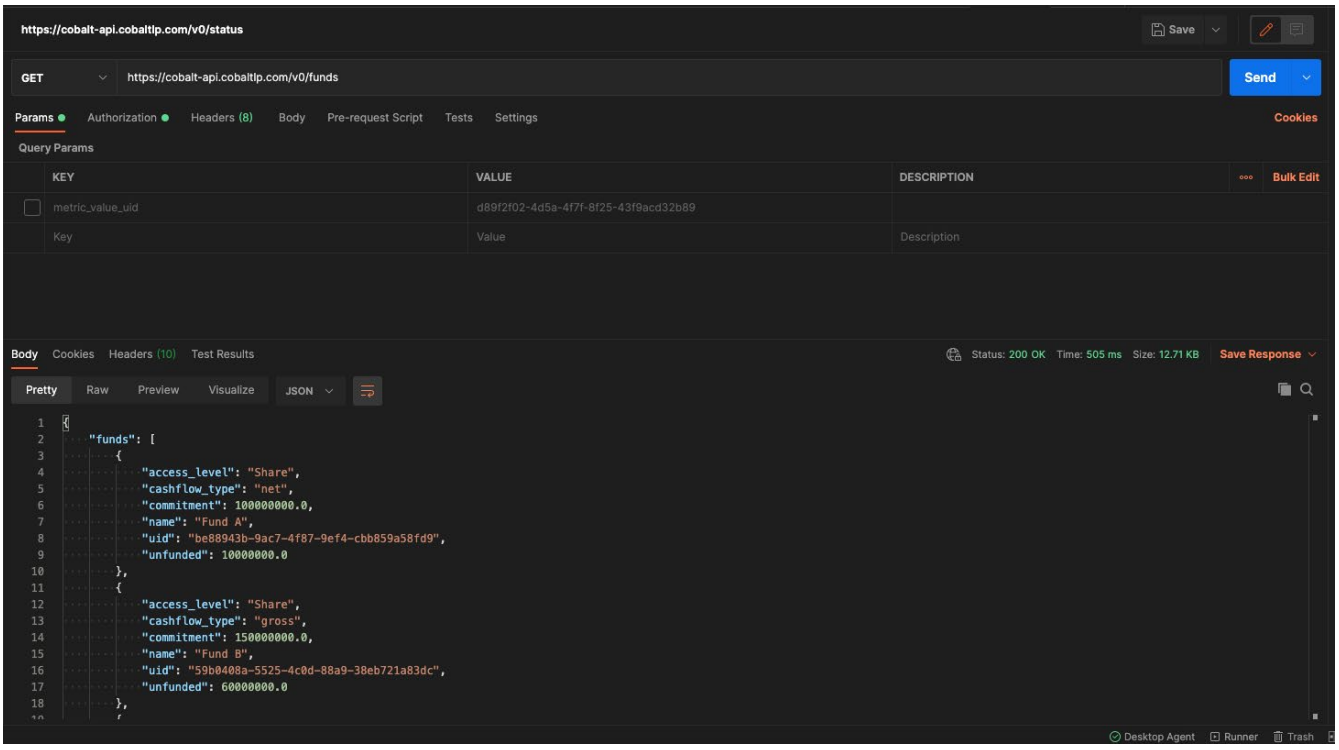
The purpose of this guide is to provide an in-depth tutorial to the Cobalt LP API through solutions to specific examples of common client API pulls and calls. In this guide, we will go over the API endpoints necessary for uploading and downloading user fund data, integrating the API with Python, and testing the API with Postman.

Reach out to your dedicated client success lead to get the necessary API key that will be used to access Cobalt's API.

Postman

Before writing code to put the API in use, it is recommended to test some endpoints with Postman. Postman is a collaborative testing platform for APIs. With it, you can call API endpoints with custom data and see the responses in real time.

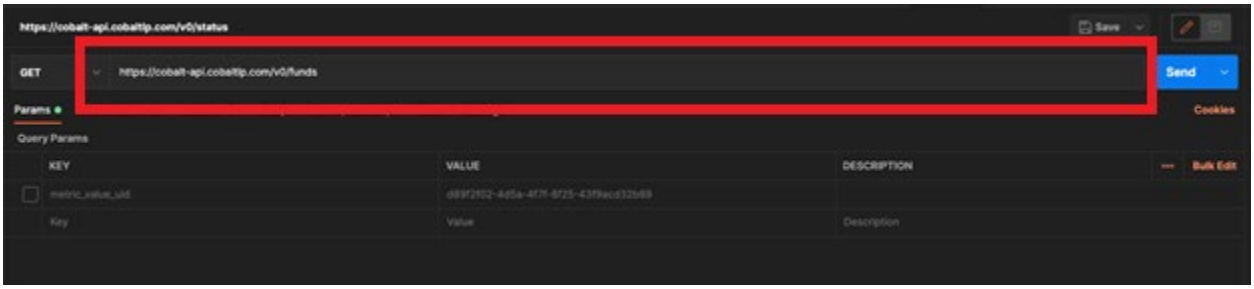
Postman also has its own detailed introduction guide available here. For the purposes of brevity, this guide will go over some of the most important parts of the screenshot below.



The URL Bar

This bar is located at the top of the image and currently says “https://cobalt-api.cobaltlp.com/v0/funds”.

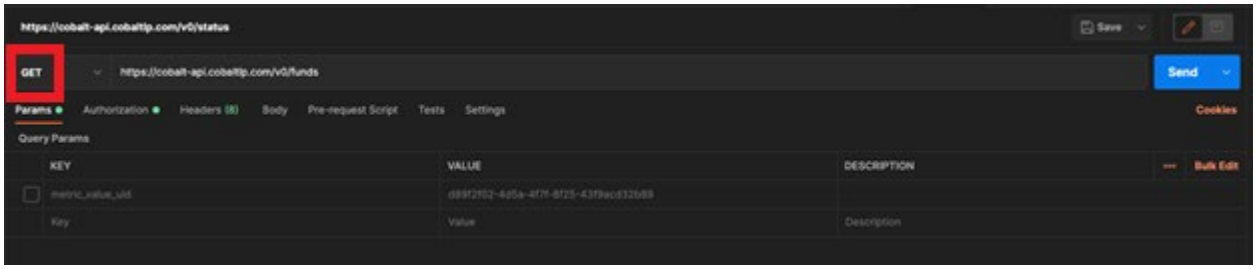
The URL bar tells Postman what part of the application you wish to perform an action on. The possible endpoints for Cobalt LP are listed in our Cobalt API Documentation.



The Method Box

This box is in the upper-left corner of the image and currently says “GET”. This will let the API know what action you want to perform on the endpoint in the URL bar. The most common options implemented are:

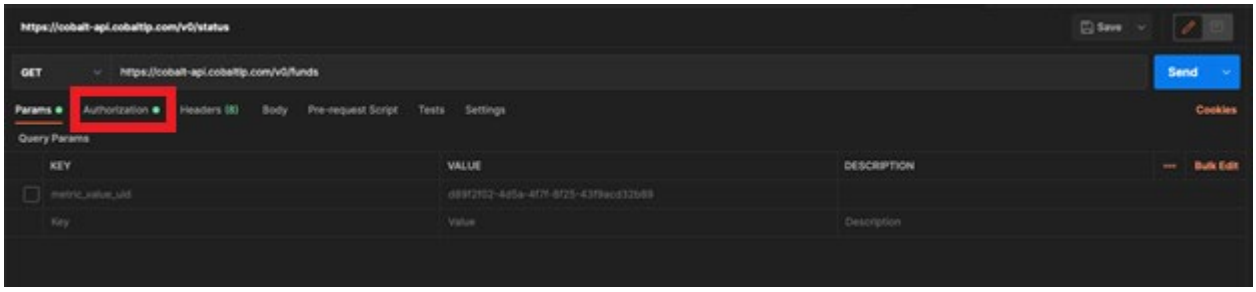
- “GET”: Retrieve data from the application
- “POST”: Create new data in the application
- “PUT”: Update data in the application
- “DELETE”: Delete data in the application



The Authorization Tab

Underneath the URL Bar, there is a tab that says “Authorization”. This tab is how Postman can connect to protected API's like Cobalt's. After clicking on the Authorization tab, you will see 4 inputs.

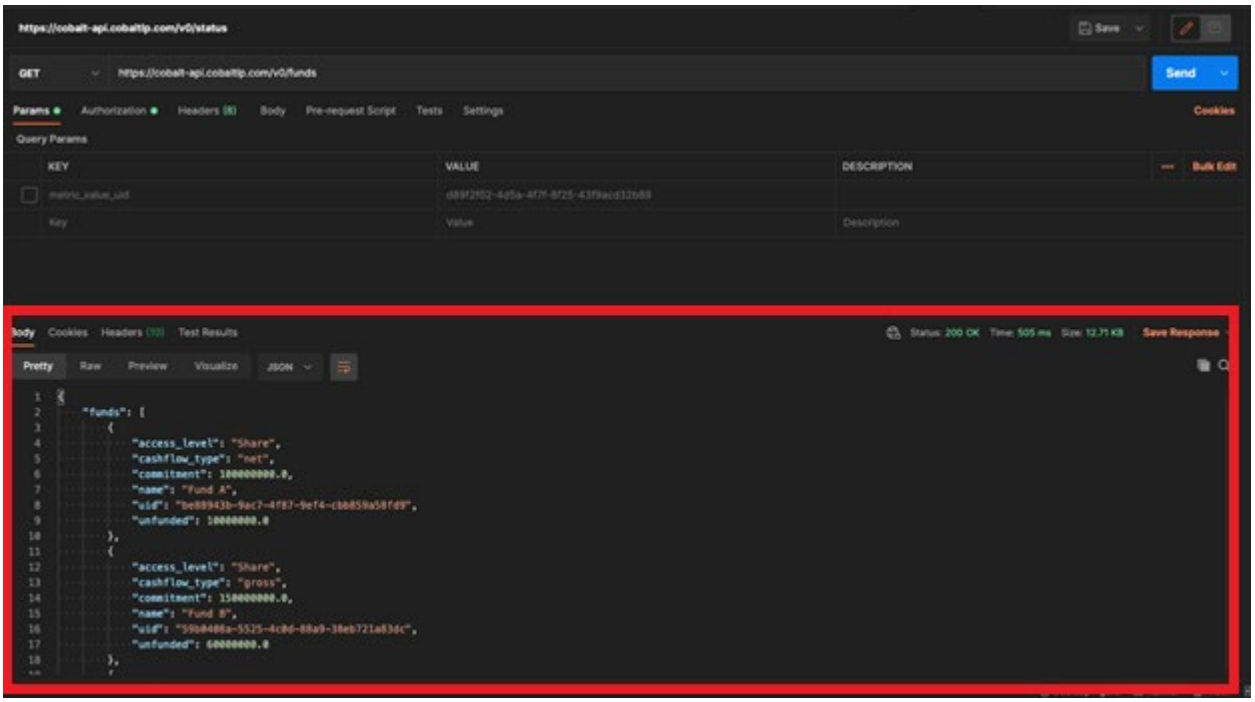
- “Type”: This should be set to “API KEY”
- “Key”: This is the header key that the application will use to look for the API key. For Cobalt, this should be set to “Authorization”
- “Value”: This is where you will copy and paste your API key that is given to you by the Cobalt LP team
- “Add to”: Make sure this is set to “Header” when using the Cobalt API



The Response Body

The response body is located at the bottom of the image below. It contains fund information that was passed back from the application after the call to “https://cobalt-api.cobaltlp.com/v0/funds” was made.

Notice the format of this information is “JSON”. It is important have a way to parse JSON while using the API within a program. Python has its own JSON library that can be accessed by using “import json”.



Python Integration

The purpose of an API is to allow other developers to speed-up and automate certain processes within the application by writing programs of their own. Almost all modern programming languages have a way of using APIs, but for this tutorial, we will focus on the programming language “Python”.

```
import requests
import json

url = 'https://cobalt-api.cobaltlp.com/v0/funds'
auth_token = 'ADD YOUR AUTH KEY HERE'

x = requests.get(url, headers={"Authorization": auth_token})

data = x.json()
print(json.dumps(data))
```

Example: Python implementation of GET https://cobalt-api.cobaltlp.com/v0/funds

Where:

- “url”: The URL of the endpoint that you want to call
- “auth_token”: This is where you will add your API key provided to you by the Cobalt LP team
- “requests.get”: The function is used to access the API and return information. More documentation on the requests library can be found here.
- “json.dumps()”: The request module has a “.json()” function that returns the response of the call in json. In order to parse this, we use “json.dumps()” to normalize the data for Python.

Uploading User Fund Data

Below, we have an example fund attributes spreadsheet. Continue reading for an example run through of the process for uploading a fund from this spreadsheet.

Entity Name	Currency	Commitment	Unfunded	Vintage Year	Vintage Year 2	Default PME Ind	Sector	Manager/GP
Fund 1	EUR	10,000,000.00	2,760,407.72	2018		MSCI ACWI IMI	Multi-Sector	Manager 1
Fund 2	USD	10,000,000.00	7,502,811.02	2019		MSCI ACWI IMI	Multi-Sector	Manager 1
Fund 3	USD	3,000,000.00	258,737.00	2009		MSCI ACWI IMI	Multi-Sector	Manager 2
Fund 4	USD	7,000,000.00	1,316,877.43	2011		MSCI ACWI IMI	Multi-Sector	Manager 3
Fund 5	USD	5,000,000.00	4,768,179.08	2020		MSCI ACWI IMI	Multi-Sector	Manager 3
Fund 6	USD	10,000,000.00	8,215,661.00	2018		MSCI ACWI IMI	Multi-Sector	Manager 4
Fund 7	USD	15,000,000.00	1,947,830.02	2015		MSCI ACWI IMI	Multi-Sector	Manager 5
Fund 8	USD	10,000,000.00	2,300,000.00	2016		MSCI ACWI IMI	Multi-Sector	Manager 6
Fund 9	USD	10,000,000.00	8,999,999.99	2020		MSCI ACWI IMI	Multi-Sector	Manager 6
Fund 10	USD	1,375,000.00	122,764.00	2019		MSCI ACWI IMI	Multi-Sector	Manager 7

Creating a Fund

First, you must create a fund using POST “https://cobalt-api.cobaltlp.com/v0/funds”. This call behaves similarly to the Python example above, with the exception that you will now need to send a request body along with the call.

Parameters

- “name”: Provide the name of the fund
- ‘cashflow_type”: The cashflow type of the fund (for this example we will be using "net"`)
- “base_currency”: The base currency of the fund (for example, USD)
- “commitment”: The commitment amount (please note that this is an optional field for uploading)
- “unfunded”: The unfunded amount (please note that this is an optional field for uploading)
- “vintage_year”: The vintage year of the fund (please note that this is an optional field for uploading)
- “share_with_client”: True/False option to share this fund with client organization at fund creation

```
url = 'https://cobalt-api.cobaltlp.com/v0/funds'
auth_token = 'ADD YOUR API KEY HERE'

body = {
        "name": "Fund 1",
        "cashflow_type": "net",
        "base_currency": "EUR",
        "commitment": 10000000,
        "unfunded": 2760407.72,
        "vintage_year": 2020,
        "share_with_client": false,
}
x = requests.post(url, json=body, headers={"Authorization": auth_token})

data = x.json()
```

Example: Python implementation of uploading Fund 1 from the spreadsheet

The data that you get back will contain a “uid”. Please make sure to add this value to a list and save for later.

Creating / Finding Your Attribute

After fund has been created, we must now attach the corresponding attributes.

```
url = 'https://cobalt-api.cobaltlp.com/v0/attributes'
auth_token = 'ADD YOUR AUTH KEY HERE'

x = requests.get(url, headers={"Authorization": auth_token})

data = x.json()
```

Example: Python implementation of GET https://cobalt-api.cobaltlp.com/v0/attributes

There are two types of attributes: standard attributes and user-created attributes. Standard attributes include ones such as “Geography” or “Style / Focus”. However, Cobalt LP also allows for user-created attributes. Generally, you should first run GET “https://cobalt-api.cobaltlp.com/v0/attributes” to see if there is already an attribute created for the value that you want to input prior to inputting a value.

Using a Standard Attribute

For this example, we will be adding a “Sector” attribute to Fund 1. To start this, we will call GET “https://cobalt-api.cobaltlp.com/v0/attributes” to find Cobalt's GICS attribute. After the call, you should see something like this in the response data:

```
{
        "client_uid": null,
        "name": "Sector",
        "scope": null,
        "uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
        "user_uid": null
},
```

Tip! You can differentiate standard attributes from user-created attributes by looking at the “user_uid” field. If null, then it is a standard attribute. Make sure to store the `uid` of the attribute that you want to use.

Creating a New Attribute

You can also create your own attribute if there is not one that is already created for you. You will have to use POST “https://cobalt-api.cobaltlp.com/v0/attributes”. This will take a request body similar to how we create a fund. The body takes a “name” and “scope” parameter. For the purposes of this spreadsheet, scope should be set to 1.

In the returned data, you should receive a “uid”. Save this in the same way that you would save a standard attribute uid.

```
url = 'https://cobalt-api.cobaltlp.com/v0/attributes'
auth_token = 'ADD YOUR API KEY HERE'

body = {
    "name": "Custom-Attribute",
    "scope": 1
}

x = requests.post(url, json=body, headers={"Authorization": auth_token})

data = x.json()
```

Creating / Finding an Attribute Member

Using a Standard Attribute

Once you have the uid of the attribute that you want to attach to your fund, search for its corresponding members using GET “https://cobalt-api.cobaltlp.com/v0/attribute-members”.

```
sector_uid = '757d3c0d-d6bf-47f3-a0b2-40417cdbc95a'
url = 'https://cobalt-api.cobaltlp.com/v0/attribute-members' + \
'?attribute_uid=' + sector_uid
auth_token = 'ADD YOUR AUTH KEY HERE'

x = requests.get(url, headers={"Authorization": auth_token})

data = x.json()
```

```
{
        "client_uid": null,
        "name": "Sector",
        "scope": null,
        "uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
        "user_uid": null
},
```


> *Tip!* You can differentiate standard attributes from user-created attributes by looking at the “user_uid” field. If null, then it is a standard attribute. Make sure to store the `uid` of the attribute that you want to use.

Creating a New Attribute

Notice that for this call, we use the uid of the “Sector” attribute that was found earlier in the URL.

> *Tip!* This is our first example of using a query parameter. More information on what calls use query parameters can be found in our Cobalt API Documentation.

After the call, you should see a list of results in the response data.


```
{
        "attribute_members": [
                {
                        "attribute_uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
                        "name": "Technology",
                        "parent_uid": null,
                        "uid": "1f1654a6-3e85-47b1-b22d-a27f87b2d480"
                },
                {
                        "attribute_uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
                        "name": "Health Care",
                        "parent_uid": null,
                        "uid": "178f7aac-f3a7-40a8-9ef6-86db36dafab1"
                },
                {
                        "attribute_uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
                        "name": "Consumer Discretionary",
                        "parent_uid": null,
                        "uid": "cd759ca5-6cf2-4dc5-9e91-64bdf1a0e746"
                },
                {
                        "attribute_uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
                        "name": "Consumer Staples",
                        "parent_uid": null,
                        "uid": "c6a77a75-ac10-4843-9ed7-9120df971879"
                },
```

```
            {
                "attribute_uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
                "name": "Energy",
                "parent_uid": null,
                "uid": "8cb52cd6-f2a6-4a7a-8067-42d9ffa6a289"
            },
            {
                "attribute_uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
                "name": "Materials and Processing",
                "parent_uid": null,
                "uid": "d20b9aa2-cbed-4a2a-a7dc-f4e781388095"
            },
            {
                "attribute_uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
                "name": "Producer Durables",
                "parent_uid": null,
                "uid": "4a2d2694-e4f3-452b-a284-6654a440cf28"
            },
            {
                "attribute_uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
                "name": "Financial Services",
                "parent_uid": null,
                "uid": "8c2ce37d-9e3c-4bb9-9935-694e6e21b072"
            },
            {
                "attribute_uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
                "name": "Utilities",
                "parent_uid": null,
                "uid": "82b6759e-1d23-4e6f-ad7c-addce26cdfaf"
            },
            {
                "attribute_uid": "757d3c0d-d6bf-47f3-a0b2-40417cdbc95a",
                "name": "Multi-Sector",
                "parent_uid": null,
                "uid": "b6591716-279b-464e-8c55-9145ef3d8a70"
            }
        ]
}
```


Find the member that you would like to attach to the “Sector” attribute of your fund. Don’t forget to save the uid for later.

Using a User-Created Attribute

If you are using a user-created attribute, chances are you will need to create a member for that attribute. This can be done using POST “https://cobalt-api.cobaltlp.com/v0/attribute-members”.

```
url = 'https://cobalt-api.cobaltlp.com/v0/attribute-members'
auth_token = 'ADD YOUR API KEY HERE'

body = {
    "attribute_uid": "The UID of your attribute goes here",
    "name": "The name of your attribute member"
}

x = requests.post(url, json=body, headers={"Authorization": auth_token})

data = x.json()
```

In the response data, you will get a “uid”. Save this for later, just like the other calls.

Attaching an Attribute to a Fund

Now that you have your attribute member uid, you are ready to attach it to a fund using POST `https://cobalt-api.cobaltlp.com/v0/attribute-values`.

The following are necessary parameters for attaching an attribute to a fund:

- “fund_uid”: The uid of the fund that you want to attach the attribute to
- “attribute_member_uid”: The uid of the attribute member that you want to attach

> *Tip!* The same process is used for adding attributes to companies and deals, more information is in the Cobalt API Documentation.

```
url = 'https://cobalt-api.cobaltlp.com/v0/attribute-values'
auth_token = 'ADD YOUR API KEY HERE'

body = {
  "fund_uid": "The UID for fund 1 goes here",
  "attribute_member_uid": "b6591716-279b-464e-8c55-9145ef3d8a70"
}

x = requests.post(url, json=body, headers={"Authorization": auth_token})

data = x.json()
```

Example: Python implementation of adding a Multi-Sector Sector attribute to Fund 1

Fund 1 will now have the Sector attribute populated with Multi-Sector.

Adding Cashflows to Funds

Compared to attributes, this is relatively straightforward. To add a cashflow, we will be using POST “https://cobalt-api.cobaltlp.com/v0/cashflows”.

The following are necessary parameters for adding cashflows to a fund:

- “fund_uid”: The uid of the fund that you want to add a cashflow to
- “amount”: The amount of the cashflow
- “cf_type”: Cashflow type (example: Net Asset Value)
- “date”: The date of the cashflow in timestamp form. Python has a date / time module that will help with this
- “note”: This is where you would add any additional information about the cashflow (please note that this is an optional field for uploading)

```
url = 'https://cobalt-api.cobaltlp.com/v0/cashflows'
auth_token = 'ADD YOUR API KEY HERE'

body = {
    "fund_uid": "The UID for fund 1 goes here",
    "amount": -82274,
    "cf_type": "nav",
    "date": 1237766400,  #Start of the day on 3/23/09 in timestamp form
    "note": "Capital Call"
}

x = requests.post(url, json=body, headers={"Authorization": auth_token})

data = x.json()
```

Example: Python implementation of adding a cashflow to Fund 1

Sharing a Fund

To share a fund, we POST to “https://cobalt-api.cobaltlp.com/v0/share/user_fund”.

The following are parameters for sharing a fund:

- “user_fund_uid”: The ‘uid’ of the fund that you want to share
- “user_uid”: The ‘uid’ of the individual to share to
- “email”: The email address (registered in Cobalt LP) of the individual to share to
- “display_name”: Name of the individual being shared to
- “read”: Read access level
- “write”: Write access level
- “share”: Share access level

```
url = 'https://cobalt-api.cobaltlp.com/v0/share/user_fund'
auth_token = 'ADD YOUR API KEY HERE'

body = {
      "user_fund_uid": "The uid of the fund to be shared",
      "user_uid": "The uid of the user to be shared to",
      "email": "first.last@domain.com",
      "display_name": "First Last (<email>@<domain>.com)",
      "read": True,
      "write": True,
      "share": True
}

x = requests.post(url, json=body, headers={"Authorization": auth_token})

data = x.json()
```

Example: Python implementation of sharing a fund

Note: If known, ‘user_uid’ is the most robust method to share. Sharing with an email will also share.

The result of this request is a copy of the share record with a ‘uid’ of the share. Note the ‘uid’ can be used in deletion if a fund is no longer needing to be shared.


```
{
      "client_uid": null,
      "display_name": "First Last (first.last@domain.com)",
      "read": true,
      "write": true,
      "shared_by_user_uid": "82eae867-8975-41fa-8e43-5677b386d8f2",
      "uid": "31f9b93f-a3a3-4121-8f3b-5fcd531d66f3",
      "user_fund_uid": "b21cae67-f7eb-473c-a34c-3cace80eaf6b",
      "user_uid": "3cf1b93e-2b99-4568-be5d-48e1233bb967",
      "email": "first.last@domain.com",
      "write": true
}
```

Sharing a Fund

To share a fund, we POST to “https://cobalt-api.cobaltlp.com/v0/share/user_fund”.

The following are parameters for sharing a fund:

- “user_fund_uid”: The ‘uid’ of the fund that you want to share
- “user_uid”: The ‘uid’ of the individual to share to
- “email”: The email address (registered in Cobalt LP) of the individual to share to
- “display_name”: Name of the individual being shared to
- “read”: Read access level
- “write”: Write access level
- “share”: Share access level

```
url = 'https://cobalt-api.cobaltlp.com/v0/share/user_fund'
auth_token = 'ADD YOUR API KEY HERE'

body = {
      "user_fund_uid": "The uid of the fund to be shared",
      "user_uid": "The uid of the user to be shared to",
      "email": "first.last@domain.com",
      "display_name": "First Last (<email>@<domain>.com)",
      "read": True,
      "write": True,
      "share": True
}

x = requests.post(url, json=body, headers={"Authorization": auth_token})

data = x.json()
```

Example: Python implementation of sharing a fund

Note: If known, ‘user_uid’ is the most robust method to share. Sharing with an email will also share.

The result of this request is a copy of the share record with a ‘uid’ of the share. Note the ‘uid’ can be used in deletion if a fund is no longer needing to be shared.


```
{
      "client_uid": null,
      "display_name": "First Last (first.last@domain.com)",
      "read": true,
      "write": true,
      "shared_by_user_uid": "82eae867-8975-41fa-8e43-5677b386d8f2",
      "uid": "31f9b93f-a3a3-4121-8f3b-5fcd531d66f3",
      "user_fund_uid": "b21cae67-f7eb-473c-a34c-3cace80eaf6b",
      "user_uid": "3cf1b93e-2b99-4568-be5d-48e1233bb967",
      "email": "first.last@domain.com",
      "write": true
}
```

Downloading User Fund Data

In this section, we will go over how to pull user fund data out of the application.

Getting User Fund IDs

The first step to getting user fund info from the app is to get the UIDs of the funds that you want to take information from. To do this, we will be using the GET “https://cobalt-api.cobaltlp.com/v0/funds”. You will not need to provide any parameters for this call.

```
url = 'https://cobalt-api.cobaltlp.com/v0/funds'
auth_token = 'ADD YOUR AUTH KEY HERE'

x = requests.get(url, headers={"Authorization": auth_token})

data = x.json()
```

This call will give you all the user funds to which you have access, along with some basic information and a UID. Make sure to save the UIDs of the funds that you want to get more information from as we will be needing them for future API calls.


```
{
    "funds": [
        {
            "access_level": "Share",
            "cashflow_type": "net",
            "commitment": 100000000.0,
            "name": "Fund A",
            "uid": "d35a4b0d-b7c7-4ebd-a665-cb54c2077b82",
            "unfunded": 10000000.0,
            "vintage_year": null
        },
        {
            "access_level": "Share",
            "cashflow_type": "gross",
            "commitment": 150000000.0,
            "name": "Fund B",
            "uid": "59b0408a-5525-4c0d-88a9-38eb721a83dc",
            "unfunded": 60000000.0,
            "vintage_year": null
        }
    ]
}
```

Getting User Fund Attributes

First, you will need to get a list of the available attributes and their UIDs. An example of this is mentioned above in the “Creating / Finding Your Attribute” section.

Once you have the uid of the fund and the attribute that you want to get from the fund, use GET “https://cobalt-api.cobaltlp.com/v0/attribute-values” to get the value.

```
attribute_uid = 'd35a4b0d-b7c7-4ebd-a665-cb54c2077b82'
fund_uid = '59b0408a-5525-4c0d-88a9-38eb721a83dc'
url = 'https://cobalt-api.cobaltlp.com/v0/attribute-values' + \
'?attribute_uid=' + attribute_uid + '&fund_uid=' + fund_uid
auth_token = 'ADD YOUR AUTH KEY HERE'

x = requests.get(url, headers={"Authorization": auth_token})

data = x.json()
```

Example: Python implementation of “Geography” for Fund B

Where `fund_uid` is the UID of Fund B and `attribute_uid` is the UID of the Geography attribute.


```
{
    "attribute_members": [
        {
            "attribute_uid": "d35a4b0d-b7c7-4ebd-a665-cb54c2077b82",
            "name": "North America",
            "parent_uid": null,
            "uid": "88b87e0e-efd3-4864-bacc-46ad29e7e6d2"
        }
    ]
}
```


Example: Python output of “Geography” for Fund B

Getting User Fund Attributes

First, you will need to get a list of the available attributes and their UIDs. An example of this is mentioned above in the “Creating / Finding Your Attribute” section.

Once you have the uid of the fund and the attribute that you want to get from the fund, use GET “https://cobalt-api.cobaltlp.com/v0/attribute-values” to get the value.

Getting User Fund Calculated Values

To get the calculated values of a user fund, all you will need is the “fund_uid” that you should have saved from earlier. The call that will be used is GET “https://cobalt-api.cobaltlp.com/v0/calculated-values”.

Similar to getting attributes, the calculated values call also works with portfolios an deals.

```
fund_uid = '59b0408a-5525-4c0d-88a9-38eb721a83dc'
url = 'https://cobalt-api.cobaltlp.com/v0/calculated-values' + \
'?fund_uid=' + fund_uid
auth_token = 'ADD YOUR AUTH KEY HERE'

x = requests.get(url, headers={"Authorization": auth_token})

data = x.json()
```

Example: Python implementation


```
{
    "as_of_date": 1498780800,
    "commitment": 150000000.0,
    "distributed": 116809122.0,
    "dpi": 2.474769553822297,
    "entity_type": "user_fund",
    "irr": 0.3969195965032448,
    "loss_ratio": null,
    "name": "Fund B",
    "nav": 0.0,
    "paid_in": 47199999.620000005,
    "rvpi": 0.0,
    "start_date": 1330992000,
    "total_value": 116809122.0,
    "tvpi": 2.474769553822297,
    "unfunded": 60000000.0,
    "vintage_year": 2012
}
```


Example: Python output

Still need help? Send an email to support@cobaltlp.com

Disclosures

This presentation has been prepared solely for informational purposes and contains confidential and proprietary information, the disclosure of which could be harmful to Hamilton Lane. Accordingly, the recipients of this presentation are requested to maintain the confidentiality of the information contained herein. This presentation may not be copied or distributed, in whole or in part, without the prior written consent of Hamilton Lane.

All opinions, estimates and forecasts of future performance or other events contained herein are based on information available to Hamilton Lane as of the date of this presentation and are subject to change. Past performance of the investments described herein is not indicative of future results. In addition, nothing contained herein shall be deemed to be a prediction of future performance. Certain information included in this presentation has not been reviewed or audited by independent public accountants. Certain information included herein has been obtained from sources that Hamilton Lane believes to be reliable but the accuracy of such information cannot be guaranteed.

This presentation is not an offer to sell, or a solicitation of any offer to buy, any security or to enter into any agreement with Hamilton Lane or any of its affiliates. Any such offering will be made only at your request. We do not intend that any public offering will be made by us at any time with respect to any potential transaction discussed in this presentation. Any offering or potential transaction will be made pursuant to separate documentation negotiated between us, which will supersede entirely the information contained herein.

The S&P 500 Total Return Index is a capitalization-weighted index of 500 U.S. large cap stocks that assumes all dividends and distributions are reinvested.

The MSCI World Index is a free float-adjusted market capitalization weighted index that is designed to measure the equity performance of developed markets.

Hamilton Lane (Germany) GmbH is a wholly-owned subsidiary of Hamilton Lane Advisors, L.L.C. Hamilton Lane (Germany) GmbH is authorised and regulated by the Federal Financial Supervisory Authority (BaFin). In the European Economic Area this communication is directed solely at persons who would be classified as professional investors within the meaning of Directive 2011/61/EU (AIFMD). Its contents are not directed at, may not be suitable for and should not be relied upon by retail clients.

Hamilton Lane (UK) Limited is a wholly-owned subsidiary of Hamilton Lane Advisors, L.L.C. Hamilton Lane (UK) Limited is authorised and regulated by the Financial Conduct Authority (FCA). In the United Kingdom this communication is directed solely at persons who would be classified as a professional client or eligible counterparty under the FCA Handbook of Rules and Guidance. Its contents are not directed at, may not be suitable for and should not be relied upon by retail clients.

Any tables, graphs or charts relating to past performance included in this presentation are intended only to illustrate the performance of the indices, composites, specific accounts or funds referred to for the historical periods shown. Such tables, graphs and charts are not intended to predict future performance and should not be used as the basis for an investment decision.

Hamilton Lane Advisors, L.L.C. is exempt from the requirement to hold an Australian financial services license under the Corporations Act 2001 in respect of the financial services by operation of ASIC Class Order 03/1100: U.S. SEC regulated financial service provides. Hamilton Lane Advisors, .L.C. is regulated by the SEC under U.S. laws, which differ from Australian laws. The PDS and target market determination for the Hamilton Lane Global Private Assets Fund (AUD) can be obtained by calling 02 9293 7950 or visiting our website www.hamiltonlane.com.au.

The information herein is not intended to provide, and should not be relied upon for, accounting, legal, tax advice or investment recommendations. You should consult your accounting, legal, tax or other advisors about the matters discussed herein.

The calculations contained in this document are made by Hamilton Lane based on information provided by the general partner (e.g. cash flows and valuations), and have not been prepared, reviewed or approved by the general partners.

As of April 15, 2024.

Contact Information

Philadelphia (Headquarters)
Seven Tower Bridge
110 Washington Street
Suite 1300
Conshohocken, PA 19428
USA
+1 610 934 2222

London
4th Floor
10 Bressenden Place
London SW1E 5DH
United Kingdom
+44 20 8152 4163

Portland
Kruse Woods II
5335 Meadows Rd Suite 280
Lake Oswego, OR 97035
USA
+1 503 624 9910

Shanghai
One ICC
Shanghai International Commerce Centre
No. 288 South Shaanxi Road
Xuhui, Shanghai Municipality 200031
+021 8012 3630

Tokyo
13F, Marunouchi Bldg.
2-4-1, Marunouchi
Chiyoda-ku
Tokyo 100-6313, Japan
+81 (0) 3 5860 3940

Denver
10333 East Dry Creek Road
Suite 310
Englewood, CO 80112
USA
+1 866 361 1720

Mexico City
Av. Paseo de la Reforma 333
Espacio de oficina 417
Cuauhtémoc, 06500
Ciudad de México, CDMX
Mexico
+52 55 6828 7930

San Diego
7817 Ivanhoe Avenue
Suite 310
La Jolla, CA 92037
USA
+1 858 410 9967

Singapore
12 Marina View
Asia Square Tower 2
Suite 26-04
Singapore, 018961
+65 6856 0920

Toronto
40 King Street W
Suite 3603
Toronto, M5H 3Y2
Canada
+1 437 600 3006

Frankfurt
Schillerstr. 12
60313 Frankfurt am Main
Germany
+49 69 153 259 93

Miami
999 Brickell Avenue
Suite 720
Miami, FL 33131
USA
+1 954 745 2780

San Francisco
201 California Street,
Suite 550
San Francisco, CA 94111
USA
+1 415 365 1056

Stockholm
Östermalmstorg 1, Floor 4
114 42 Stockholm
Sweden
+44 20 8152 4163

Zürich
Hamilton Lane (Switzerland) AG
Genferstrasse 6
8002 Zürich
Switzerland
+41 (0) 43 883 0352

Hong Kong
Room 1001-3, 10th Floor
St. George's Building
2 Ice House Street
Central Hong Kong, China
+852 3987 7191

Milan
Via Filippo Turati 30
20121 Milano
Italy
+39 02 3056 7133

Scranton
54 Glenmaura National Blvd
3rd Floor Suite 302
Moosic, PA 18507
USA
+1 570 247 3739

Sydney
Level 33, Aurora Place
88 Phillip Street
Sydney NSW 2000
Australia
+61 2 9293 7950

Las Vegas
3753 Howard Hughes Parkway
Suite 200
Las Vegas, NV 89169
USA
+1 702 784 7690

New York
610 Fifth Avenue, Suite 401
New York, NY 10020
USA
+1 212 752 7667

Seoul
12F, Gangnam Finance Center
152 Teheran-ro, Gangnam-gu
Seoul 06236
Republic of Korea
+82 2 6191 3200

Tel Aviv
6 Hahoshlim Street
Building C 7th Floor
Hertzelia Pituach, 4672201
P.O. Box 12279
Israel
+972 73 2716610