The Addepar API

Contents

API

Addepar is an open platform that facilitates the secure exchange of data with other applications or products through a standards-based API.

- *Getting Started with the API*
- *API Access and Authentication*
- *API Features*
- *API Endpoints*
- *API and Power Query*

The Addepar API is completely configurable and provides access to data via HTTPS using unique API endpoints.

You can tailor an integration with any application your firm is using, including:

- Re-balancers or other order management systems
- Billing applications
- Customer relationship management (CRM) applications
- General ledgers
- Data warehouses
- *Microsoft Excel via Power Query*

The Addepar API provides programmatic access to data in *JSON format* for:

- *Portfolios* associated with specific saved analysis views
- *Entities*, including both asset owners and owned investments
- *Files* and the portfolios, groups, and assets associated with each
- *Groups* of client portfolios and their associated entities
- *Users*, including user credentials and the data each user has permission to access

You can also use the *Portfolio API* to programmatically export transactions as well as analysis views. Each data set is based on a single portfolio and a saved view in Analysis or Transactions. In addition to *JSON*, you can download portfolios as CSV, TSV, or Excel files. For automated scripts, it's recommended that you use the *Jobs API* to download the results any time within 24 hours of the initial request.

If you like, you can download the *API developer documentation* included in this online guide.

Getting Started with the API

Begin by confirming you have access to the API. To do this, click your email address at the top right, select Firm Settings. If "API Access Key" is available on the left, then you can proceed with *setting up API access*.

If it's not available, you can request access from your firm adminstrator. If your firm as a whole doesn’t have access, you need to sign the Addepar API Addendum.

If your Addepar contract began after August 2016, then the addendum was included in the contract.

If not, your firm must use the *contact Support* to request a license addendum to use the Addepar API. It can take up to 5 business days for Addepar to process your addendum and grant API access. Once access is granted, you can find your API Key and Secret and provide access to the appropriate team members.

API Access and Authentication

All connections to the Addepar API must be authenticated, both through proper credentials and permissions, and through properly structured requests.

To establish API access, you must:

- Assign the user appropriate *API permissions*
- *Create an API key/secret combination* for the API user
- Ensure that each API request:
  - Includes the *required headers and follows the correct format*
  - Includes the correct *API URL and versioning*
  - Adheres to Addepar's *rate limiting requirements*

API User Permissions

To manage API integrations, your Addepar user credentials must be permissioned for API access.

Addepar API access permission includes all API-related features, including:

- Creating API keys
- Generating API access URLs that can be used to establish integrations with applications like Microsoft Power Query
- Accessing all data permissioned to the API key holder

To grant API access:

1. Navigate to your Users & Permissions settings (click your email address at the top right, select Firm Settings, and then click Users & Permissions on left).
2. From the list of firm users, choose the individual whose permissions you'd like to set.
3. Click the Tool Permissions tab.
4. Scroll down to API Access and select Full Permission.

To secure your API integrations, Addepar recommends:

- Only granting API access to users who are actively building or managing the integrations.
- Taking precautions to prevent changes to the analysis and transactions views that are used in to export portfolio data via the API. Any changes to these views could break existing integrations.
  - One method for securely managing API access is to create a dedicated API user profile in Addepar. As you are developing your integrations, you can grant the dedicated profile API access permission, as well as permission for the analysis and transactions tools and a small set of representative data to test the integrations. When a user logs in with the API profile, he or she can create personal views for each desired integration and then generate API URLS to establish the connections.
  - Once the integrations have been established and tested with trial data, the API user profile should be used only to monitor API usage. Limiting the use of the API profile protects the views used in integrations from accidental changes by other users.

API Key and Secret

API access keys are authentication credentials for an API connection. Each Addepar API key is paired with a secret shared only with its creator. Both the key and the secret are required to authenticate API requests.

Each API key and secret combination respects the tool and data permissions granted to the API key creator Addepar. In other words, your API key grants access to:

- All data you have permission to access, including both client portfolios and groups
- All tool permissions assigned to you in Addepar, including the ability to view, create, update, and delete clients, investments, groups, attributes, files, and Addepar user profiles

To protect your firm’s data, Addepar recommends taking the following precautions:

- Store each API key/secret combination in a secure location and do not share it with any other user: guard the combination with the same care you would use for any sensitive password.
- Create a different key for each integration. Doing so will protect existing integrations if a key is lost, and will also help to track who is managing each integration and how often.

- Create a separate profile with the appropriate permissions any time you would like to share a key with a third party (for example, to support an integration).
- Appoint a firm admin to monitor the API keys in use and to delete any obsolete keys

> **Tip:** To review API key usage, select "Display all access keys" in the API Access Key settings, and then review the “last used on” date for each API key.

To create an access key and secret pair:

1. Navigate to your API Access Key settings (click your email address at the top right, select Firm Settings, and then click API Access Key on left).
2. Click the plus button in the rightmost corner of the table header.
3. Enter a description of the key that explains its purpose (typically the name of the integration it supports).
4. Click Submit.
5. Record the key and secret, and store the combination in a secure location.

> **Important:** You won't be able to access the secret for this API key again once you have clicked the done button. The API secret is only displayed when the API key is created, and is salted and hashed before it is stored in the Addepar database.

6. Click Done.

Store each API key/secret combination in a secure location.

> **Important:** API keys & secrets should never be shared in publicly accessible areas such GitHub, client-side code, and so forth. Sharing as plain text in an email is also not recommended.

API Authentication Requirements

Addepar allows only authenticated requests to the API.

All API requests must be made over HTTPS. Requests made over HTTP or without authentication will fail.

The Addepar API uses *HTTP Basic Auth* to authenticate API requests. You need to pass your authentication credential as a header in each API request. The authentication credential can be constructed by:

1. Combining your username (API Key) and password (API Secret) with a single colon (:).
2. Encoding the combined string using a variant of *Base64*.
3. Prepending “Basic” and a space to the result.

> **Note:** cURL requests can include only the combined string of [Key:Secret]. cURL will automatically encode it.

To identify your firm when making API requests, you need to include your firm ID as a header. You can find your firm ID by generating an API URL in the application. To do so, open the Analysis Tool, click Export above the table, and select “Generate API URL.” Your firm ID is listed as the value of “addepar_firm=” in the URL string.

You will need to replace “firmdomain” to match the URL your firm uses to log into the Addepar application. For example, if your firm is Terra Bella Capital, your API URL may be https://terrabella.addepar.com/api/v1.

Required headers

GET and DELETE requests:

- **Authorization:** "Basic [Base64-encoded Key:Secret]"
- **Addepar-Firm:** "Firm ID"

POST and PATCH requests:

- **Authorization:** "Basic [Base64-encoded Key:Secret]"
- **Addepar-Firm:** "Firm ID"
- **Content-Type:** "application/vnd.api+json"

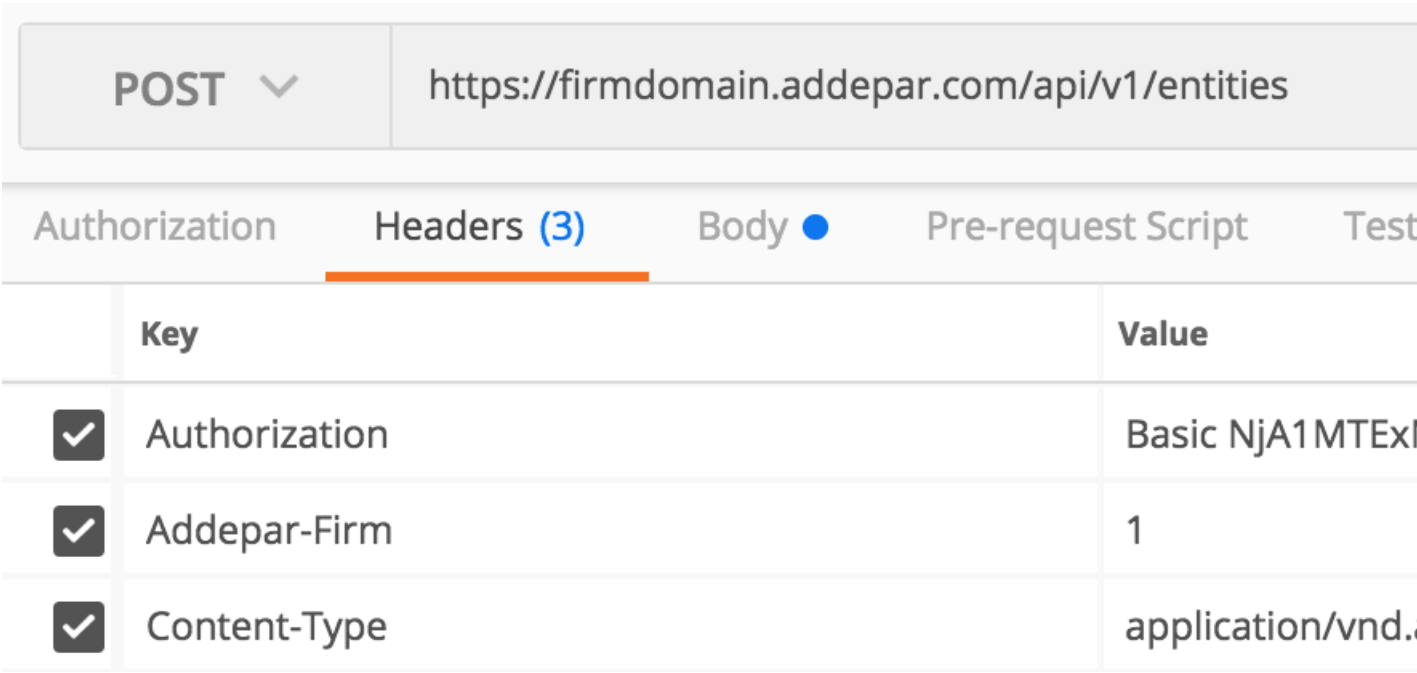
Sample Request Headers

The following examples show examples of the headers required to create, read, update, and delete various endpoints of the Addepar API. You can make requests using cURL or an HTTPS client to test and validate headers.

GET request:

```
curl -u
      36bf3dd1-d714-469b-
a9a6-3eb172dd9f73:K5JdSTip9ZcXFrG3p3jKsukskjm2NDqs6kdfAyCI -H
      "Addepar-Firm:1111" https://firmdomain.addepar.com/api/v1/users
```

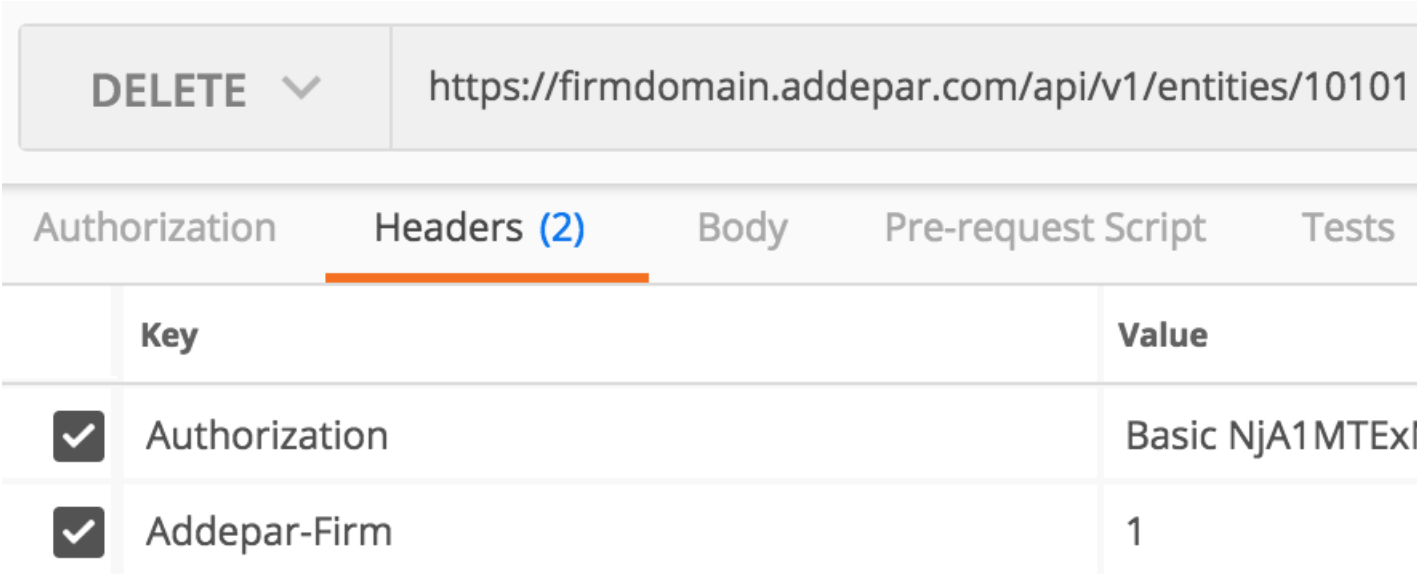
POST request:



PATCH request:

```
curl -u
      36bf3dd1-d714-469b-
a9a6-3eb172dd9f73:K5JdSTip9ZcXFrG3p3jKsukskjm2NDqs6kdfAyCI -H
      "Addepar-Firm:1111" "Content-Type:application/vnd.api+json"
      https://firmdomain.addepar.com/api/v1/entities/10101
```

DELETE request:



API Features

The Addepar API includes features to ensure the flexibility, stability, and performance of your integrations.

- *Pagination* ensures that multiple objects can be exported in a single response for "Get All" routes.
- *Rate limiting* controls the number of requests to ensure a reliable quality of service.
- *Versioning* guarantees that any time Addepar makes changes to the API that are not backward-compatible, we will release a new version and provide a backward-compatible patch.
- *Standardized API response codes* provide insight into the status of each request.

API Pagination

The Addepar API supports pagination for "Get All" routes so that multiple objects can be exported in a single response.

Addepar strongly recommends pagination to ensure reliable and consistent API performance because it prevents the result set size from increasing.

The Addepar API enforces the maximum page size of 500 results per response. If the complete results can't fit in a single response, the set will be automatically truncated and the remaining values will be linked from "next" URL appended to the end of the set.

Parameters

Parameter	Description	Type
page[limit]	The number of results to be returned. The maximum result set size is 500.	Integer

Links

Name	Description	Type
next	The URL of the "next" page of the result set. Null if there are no additional pages.	String

Example

```
# Request
GET /v1/entities?page[limit]=100


# Response
HTTP/1.1 200
{
  "data": [
    ...
  ],
  "links": {
    "next": "/v1/entities?page[limit]=100&page[after]=10001"
  }
}
```

API Rate Limiting

To ensure a reliable quality of service for all our clients, the Addepar API enforces rate limiting on API requests.

Rate limiting is implemented at the firm level and is counted over a 24 hour window for all authenticated calls.

Type	Limit
Maximum number of requests in 15 minute window	50
Maximum number of requests in 24 hour window	1000
Maximum runtime per request	60 sec

If the API request breaches a limit, the request will be rejected with “429 Too many requests” response code. The response will also contain “Retry-After” header to indicate how long an API client should wait (in seconds) before making another request. If the API request takes longer than 10 seconds, the request will be cancelled with “400 Bad Request”.

API URL and Versioning

Any time Addepar makes changes to the API that are not backward-compatible, we will release a new version. This API version number must be included in the API URL.

For backward-compatible changes to the API, Addepar will provide a new patch version via the API.

Current API Version

The current version is v1.5

API URL

You can make a GET request using the API URL to check the current API version.

```
https://firmdomain.addepar.com/api/v1/api_version
```

Note: You will need to replace [firmdomain] to match the url your firm uses to log in to the Addepar application. For example, if your firm is Terra Bella Capital, your API URL might be https://terrabella.addepar.com/api/v1/api_version

Synchronous Request URL

```
GET /api/v1/api_version
```

cURL sample request

```
curl https://firmdomain.addepar.com/api/v1/api_version
{
  "data": {
    {
        "id": "1.1",
        "type": "version",
        "attributes": {}
      }
}
```

API Response Codes

The Addepar API returns HTTP response codes to clarify the status of each API request.

Successful Request

A successful request to the Addepar API will result in one of the following status codes.

Status Code	Title	Request Type
200	OK	Successful GET or PATCH request
201	Created	Successful POST request
202	Accepted	Successful POST request to jobs route
204	No Content	Successful DELETE request Successful POST, PATCH, or DELETE to a relationships sub-route
303	See Other	Completed asynchronous job. Follow the link in the "Location" header to retrieve the results.

Unsuccessful Request

A failed request to the Addepar API will result in one of the following status codes. Please review the "detail" field in the error response for additional information about a particular failed request.

Status Code	Title	Reason	Troubleshooting
400	Bad Request	Improperly formatted query parameters or payload	Check the fields indicated in the error response.
401	Unauthorized	Failed authentication	Check the API key, secret, and "Addepar-Firm" header.
403	Forbidden	User lacks required permission(s)	Check that the user has Public API permission and Tool Permissions required for the particular URL.
404	Not Found	Incorrect URL or ID	Check that the ID is correct and the user has permission to access it.
405	Method Not Allowed	Unsupported HTTP method	Generally only GET, POST, PATCH, and DELETE are supported.

Status Code	Title	Reason	Troubleshooting
409	Conflict	Action would result in an invalid data state	Check that the "type" field matches that of the URL.
410	Gone	The resource was available but has expired	Try making another request to the Portfolio Job API.
415	Unsupported Media Type	Invalid Content-Type header	Attach "Content-Type" header "application/vnd.api+json".
429	Too Many Requests	Request rate limit has been exceeded	Check the "X-RateLimit-Retry-After" header for the number of seconds until the next request can be made.
500	Internal Server Error	An unexpected error occurred	Please contact Support if you see this.

API Security Attributes

The following security attributes apply to the Entities API, Groups API, and Positions API. These attributes can be applied to all entity types, groups, and positions, unless otherwise noted.

Time-Varying Attributes

All time-varying attributes have three required fields: Date, Value, and Weight. Each field must be present, regardless of whether the value of each of the fields is null or non-null.

When the value of the Date field is null, it means the Value and Weight fields apply at all points in time. When multiple dates are included for an attribute, it means the values of Value and Weight apply as of that date.

The value of the Weight field must be a decimal. Non-decimals will cause an error. For example, use “1.0” instead of “1”.

Some attributes can have multiple values on the same date. To represent this, the Weight field takes on a decimal value or value less than one. For example, if a stock is listed in two sectors on the same date, and the stock is divided between the two sectors 50/50, the value of each sector’s Weight is “0.5”.

The example below illustrates the different ways a time-varying attribute may appear. In this example, Asset Class, Country, and Sector are used to illustrate:

```
"attributes": {
    "asset_class": [
        {
          "date": null,
          "value": "Equity”,
          "weight": 1.0
        }
      ],
    “country”: [
        {
           "date”: “2015-01-01”,
           “value”: “USA”,
           “weight”: 1.0
        },
        {
           "date”: “2018-01-01”,
           “value”: “CAD”,
           “weight”: 1.0
        }
      ],
    "sector: [
```

```
        {
           "date”: “2018-01-01”,
           “value”: “Technology”,
           “weight”: 0.5
        },
        {
           "date”: “2018-01-01”,
           “value”: “Large Cap”,
           “weight”: 0.5
        }
    ]
}
```

Money Value Attributes

All money value attributes have two required fields: Value and Currency. Both fields must be present.

The example below illustrates how the money value Call Price attribute appears:

```
"attributes": {
    "call_price": {
        "value": "100”,
        "currency": “USD”
    }
}
```

Attribute	Description
asset_class	Asset Class. Time-Varying. Example: "Fixed Income"
bond_frequency	Frequency. String. Allowed values: • ONE_DAY • ONE_WEEK • ONE_MONTH • THREE_MONTHS • SIX_MONTHS • ONE_YEAR
bond_type	Bond Type. String. Allowed values: • CORPORATE_BOND • CONVERTIBLE_BOND • GOVERNMENT_BOND • TREASURY_BILL • TREASURY_NOTE • MUNICIPAL_BOND • TREASURY_BOND • TIPS
call_price	Call Price. Money Value. Example: "1020, USD"

callable_date	Callable Date. Date. Example: "2007-01-21"
conversion_minimum	Automatic Conversion Minimum Amount. Money Value. Example: "1200, USD" Note: Not applicable to positions or groups.
country	Country. Time-Varying. Example: "Canada, 1.0"
coupon_rate	Coupon Rate. Percent, represented as decimal. Time-Varying. Example: "0.0375"
currency_factor	Currency. String. Example: "USD"
cusip	CUSIP. Time-Varying. Example: "037833100"
dated_date	Dated Date. Date. Example: "2007-01-21"
day_count_convention	Day Count Convention. String. Allowed values: • ACTUAL_360 • ACTUAL_ACTUAL • ACTUAL_ACTUAL_ICMA • ACTUAL_365_NO_LEAP_YEAR • ACTUAL_365 • ACTUAL_365_ACTUAL • THIRTY_360_ISDA • THIRTY_360_ALTERNATIVE_EOM_CONVENTION • THIRTY_360_MSRB • THIRTY_360_SIA • THIRTY_360_US_NASD • THIRTY_360_US • THIRTY_E_360 • THIRTY_E_360_ISDA • THIRTY_E_PLUS_360
delivery_price	Delivery Price. Money Value, Currency. Example: "80.02, USD" Note: Not applicable to positions or groups.
dividend_rate	Dividend Rate. Percent, represented as decimal. Example: "0.0375"

expiration_date	Expiration Date. Date. Example: "2007-01-21"
first_payment_date	First Payment Date. Date. Example: "2007-01-21"
interest_rate	Interest Rate. Percent, represented as decimal. Example: "0.0375"
investment_type	Investment Type. String. Example: "Bond"
is_callable	Callable. String. Allowed values: • CALLABLE • NOT_CALLABLE
is_cumulative	Cumulative. String. Allowed values: • CUMULATIVE • NOT_CUMULATIVE
is_prerefunded	Prerefunded. Boolean. Allowed Values: • TRUE • FALSE
is_rolled_up	Rollup. Boolean. Allowed Values: • TRUE • FALSE Note: Not applicable to clients, managed funds, positions, or groups.
isin	ISIN. Time-Varying. Example: "US0378331005"
issue_date	Issue Date. Date. Example: "2007-01-21"
liquidation_return	Liquidation Return. Number. Example: "3.2"
maturity_date	Maturity Date. Date. Example: "2007-01-21"

multiplier	Multiplier. Number. Example: "4"
no_lookthrough	Disable Lookthrough. Boolean. Allowed Values: • TRUE • FALSE Note: Can only be applied to managed and private funds. Not applicable to positions or groups.
node_strike_price	Strike Price. Number. Example: "4.26"
node_yield	Current Yield. Percent, represented as decimal. Example: "0.0375"
note_discount	Note Discount. Percent, represented as decimal. Time-Varying. Example: "0.0375"
option_status	Option Status. Time-Varying. Allowed values: • GENERIC • ISO • NSO Note:
option_type	Option Type. String. Allowed values: • CALL • PUT
original_principal_per_share	Original Principal Per Unit. Number. Example: "113.82"
projected_annual_income	Projected Annual Income (Per Unit). Number. Example: "3210"
sector	Sector. Time-Varying. Example: "Technology"
sedol	SEDOL. Time-Varying. Example: "2046251"

settlement_type	Settlement Type. String. Allowed values: • CASH • PHYSICAL
ticker_symbol	Ticker Symbol. Time-Varying. Example: "AAPL"
valuation_cap	Valuation Cap. Money Value. Time-Varying. Example: "1000, USD"

API Endpoints

Once you've established API access, you can access Addepar data via API endpoints.

API Endpoints include:

- *Portfolios*
- *Entities*
- *Groups*
- *Files*
- *Users*
- *Jobs*

Portfolio API

You can use the Portfolio API to automate a process to export portfolio data from Addepar, including both analysis and transactions data.

The JSON output respects nested groupings, meaning each portfolio you retrieve in JSON will include the same grouping structure that you've configured in each view. JSON output is currently only available for analysis data.

Each set of data you export with the API is based on a saved view populated with data from a specific portfolio. You can use the API to download analysis views in JSON format to integrate with other systems or analysis and transactions views as a CSV, TSV, or Excel file to perform additional analysis.

You can also generate an API URL for portfolio views in Addepar to use in establishing an integration with an external application like Microsoft *Power Query*.

Tip: To download large analysis views, you can make an asynchronous request to the Portfolio Jobs API.

Summary

Base Route	/v1/portfolio/views/:view-id/results /v1/transactions/views/:view-id/results
Methods	GET
Output Format	*JSON*, CSV, TSV, or XLSX
Pagination	No
Permissions Required	Analysis: View-only or Full permission Transactions: Files View-only or Full permission

Arguments

Argument	Description	Values
view-id	ID of a saved Analysis or Transactions view	You can find the view ID by generating a Portfolio API URL (see instructions below). The view ID is following the "/views/" section of the URL. For example, view ID 15 will be noted as "/views/15".

Parameters

Parameter	Definition	Values*
portfolio_id	The ID of a portfolio configured in Addepar. A portfolio can be either an entity (i.e. a client, account, legal entity, security etc.) or a group of entities.	The ID number Note: If "portfolio_type" is "firm", the "portfolio_id" must be "1."
portfolio_type	Type of the portfolio_id portfolio. Allowed values: 'group', 'entity' or 'firm'.	"group", "entity", or "firm"
start_date	The start date of the reference period to be used for the analysis view.	A date in yyyy-MM-dd format
end_date	The end date of the reference period to be used for the analysis view.	A date in yyyy-MM-dd format
output_type	Output format of the exported result.	"json", csv", "tsv", or "xlsx"

* You can find the values for the ID, type, and dates by generating a Portfolio API URL for the desired view and portfolio.

Outputs

JSON format provides an easy way to integrate portfolio data into other systems with minimal maintenance. JSON output is currently only available for analysis data.

CSV and TSV formats may be reformatted to enable more specific analysis, but entail view configuration restrictions.

Analysis and Transactions Views

- **Entity Values.** When the output includes a column that holds entity values (for example, security, top level owner, or direct owner), the output will also include an "[Entity Name] [Entity ID]" column to record the unique Addepar IDs for each entity.
- **Positions.** If the position attribute is included as an output column, an additional column will be added to include the Addepar unique ID for each position. The header of this column will use the same value as previous column appended with the text "[Position ID]".
- **Formatted numerical values.** Formatting applied to numerical values (for example $, K, M etc.) is ignored in the output for number and money value columns: the output will include the raw values, exported with a precision of up to 4 decimal places.
- **Percentage values.** Percentages are represented as a fraction, with precision up to 4 decimal places (e.g. 5.1% will be 0.051).
- **Boolean Values.** Values of Boolean (yes/no) attributes are represented as "true" or "false" in the API output.
- **Byte-Order-Mark.** All CSV/TSV responses have the Byte-Order-Mark prefixed to the data stream to indicate the UTF-8 encoding of the content.

  Refer to Wikipedia for more information about Byte-Order-Mark.

Portfolio API URLs

Portfolio API URLs can be used by external applications like Microsoft Excel Power Query to access portfolio data from an Analysis or Transactions view. The API URL also includes ids for the firm, view, and portfolio that you can use in API requests.

To generate an API URL, select the portfolio from which you'd like to export data and open an Analysis or Transactions view configured to include the information required by your integration. Then click the Export button above the table and choose "Generate API URL." Copy the displayed link by clicking control+c or clicking the "Copy Link" button.

The URL includes:

- View id ("/views/15")
- Portfolio id ("portfolio_id=1")
- Firm id ("addepar_firm=1")

You can also use the API URL to establish an integration. For example, if you're using Power Query, you can connect the Addepar API to Excel in a *few simple steps*.

Analysis Views

JSON

- **Grouping-Level Values.** All groupings are included in the output.
- **Empty Groupings.** If the asset table includes no groupings, the output will include a "TOTAL" field that includes the total portfolio value.

CSV, TSV, and XLSX

- **Grouping-Level Values.** Only data from the lowest-level grouping is included: higher level groupings and rollup values are omitted. For example, if your asset table is grouped by Asset Class > Security, only security values will appear in the output.
- **Empty Groupings.** If the asset table includes no groupings, the output will include a "TOTAL" column that includes the total portfolio value.

Analysis Data

API output from the Analysis tool is based on the asset table configuration of a saved view.

Common API Analysis Views

The following are a few Analysis view configurations that provide sets of data commonly required for an integration with Addepar:

- All clients: Group securities by Top Level Owner.
- All accounts associated with all clients: Group by Top Level Owner, then Direct Owner. Include a column for Top Level Owner.
- All Security Holdings: Group by Direct Owner, then Security. Include columns for Position and Direct Owner to automatically add the Direct Owner's Position and Entity IDs of the to the API exports.
- All Securities: Group by Securities.

Example

```
# Request
GET https://examplefirm.com/api/v1/portfolio/views/1/results?
portfolio_id=10&portfolio_type=entity&output_type=csv&start_date=2016-01-01&end_date=201


# Response
HTTP/1.1 200

<VIEW_CONTENT>
```

Restrictions on JSON Output for Analysis Views

- JSON format supports groupings at all levels.
- Views with “Advanced” and “Pivot” tables are not supported.
- Column values that dash or are null in the application are not included in the JSON output.
- Exported attributes must be one of the following types: Currency, Date, List, Money Value, Number, Percent, Word, Yes/No.

Restrictions on CSV/TSV Output for Analysis Views

- Views with “Advanced” and “Pivot” tables are not supported.
- Benchmark rows are not exported.
- Graphs and charts cannot be exported.
- The holding account, ownership structure, and legal entity attributes are not allowed at the lowest grouping level: attempting to export a view with such a configuration will return a 400 Bad Request response.
- Exported attributes must be one of the following types: Currency, Date, Money Value, Number, Percent, Word, Yes/No.

Analysis Data Job

For longer running Analysis, it is recommended to use the Jobs API to submit an asynchronous request and fetch the results later. To submit the same request above as a job, format the request as shown below. See the *Jobs API* for information about checking the status of the job and downloading the results.

Example

```
# Request
HTTP/1.1 200
Content-Disposition: attachment; filename="portfolio_data.csv"
Content-Type: text/csv

<VIEW_CONTENT>

# Response
HTTP/1.1 200

{
  "data": {
    "id": "14",
    "type": "jobs",
    "attributes": {
      "job_type": "PORTFOLIO_VIEW_RESULTS",
      "parameters": {
        "view_id": 1,
        "portfolio_type": "entity",
        "portfolio_id": 10,
        "output_type": "csv",
        "start_date": "2016-01-01",
        "end_date": "2016-01-02"
      },
      "status": "Queued"
    },
    "links": {
      "self": "/v1/jobs/14"
    }
  }
}
```

Transaction Data

API output from the Transactions tool is based on a saved configuration of the transactions table.

Example

```
# Request
GET https://examplefirm.com/api/v1/transactions/views/1/results?
portfolio_id=10&portfolio_type=entity&output_type=csv&start_date=2016-01-01&end_date=201

# Response
HTTP/1.1 200
Content-Disposition: attachment; filename="transaction_data.csv"
Content-Type: text/csv

<VIEW_CONTENT>
```

Restrictions on CSV/TSV Output for Transaction Views

- Transaction views for “Summary Data” are not supported.
- Exported attributes must be one of the following types: Currency, Date, Money Value, Number, Percent, Word, Yes/No.

Entities API

The Entities API resource can be used to create, read, update, and delete financial entity data in Addepar.

Entities include all asset owners and owned investments: for example, clients who own portfolios, legal entities, accounts, and assets. Each entity is described with a set of attributes, which specify how the client or investment is represented in Addepar.

Summary

Base Route	/v1/entities
Methods	GET, POST, PATCH, DELETE
Output Format	*JSON*
Pagination	Yes
Permissions Required	• View, edit, and delete: "Access to curent and future portfolios" • Create: N/A

Attributes

The API supports both standard attributes (those provided by Addepar) and custom attributes (those defined and created by your firm). All attributes, whether standard or custom, are grouped in the "attributes" component of the resource object.

Standard Attributes

Standard attributes include descriptive details about portfolio owners, like the client’s name, and investments, like a bond’s issue date or a security’s CUSIP.

Supported standard attributes are listed below.

Name	Description
original_name	Name. String. Required. Example: "Adam Smith"

Name	Description
display_name	Name given to the entity. String. Note: Cannot be applied to a client.
model_type	String. Required. Allowed values: • BOND • CASH • CERTIFICATE_OF_DEPOSIT • CLOSED_END_FUND • CMO • CONVERTIBLE_NOTE • ETF • ETN • FINANCIAL_ACCOUNT • FORWARD_CONTRACT • GENERIC_ASSET • GENERIC_COMPANY_GRAPH_NODE • HOLDING_COMPANY • MASTER_LIMITED_PARTNERSHIP • MONEY_MARKET_FUND • MUTUAL_FUND • OPTION • PERSON_NODE • PREFERRED_STOCK • REIT • STOCK • TRUST • UIT • UNKNOWN_SECURITY • WARRANT
ownership_type	String. Not editable. Allowed values: • PERCENT_BASED • SHARE_BASED • VALUE_BASED Note: This attribute does not apply to a client.
[security attribute]	All *security attributes* can be applied to groups, unless otherwise noted.

Custom Attributes

Custom attributes are identified in the API with the prefix "_custom_" (to easily differentiate between standard and custom attributes), followed by the attribute's given name, and finally its Addepar attribute ID (to differentiate it from other, similarly named custom attributes):

```
"_custom_<Attribute Name>_<Attribute ID>"
```

For example, a custom attribute named "My Asset Class" with ID "1234" will be named "_custom_my_asset_class_1234" in the API.

Outputs

Output Types

The API supports standard and custom attributes that hold any of the following Addepar output types:

Output Type	Description	Example
Word	A string	`"Equities"`
Number	A number	`100`
Percentage	A percentage expressed as decimal	`0.05`
Date	A date in YYYY-MM-DD format	`"2017-03-30"`
Yes/No	A boolean (true or false)	`true`
Currency	Currency code as a string	`"USD"`
Money Value	An object containing currency and value	`{"currency": "USD", "value": 167.23}`

Time-Varying Attributes

Some standard attributes, and all custom attributes, support time-varying outputs to reflect changes in value over time (for example, to record a change in a stock's geography by holding different values before and after a merger or another event). For more information on time-varying attributes, see *Security Attributes*.

Restrictions

You can only create (POST) and update (PATCH) the following entity types using the Entities API:

Name	Model Type
Client	PERSON_NODE
Trust	TRUST
Holding Company	HOLDING_COMPANY
Financial Account	FINANCIAL_ACCOUNT

Retrieve All Entities

```
GET /v1/entities
```

Retrieve the full list of all entities the authenticated user has permission to access, as well as all available attribute details for each entity.

Example

```

# Request
```

```
GET https://examplefirm.com/api/v1/entities


# Response
HTTP/1.1 200

{
  "data": [
    {
      "id": "1000001",
      "type": "entities",
      "attributes": {
        "currency_factor": "USD",
        "original_name": "Adam Smith",
        "model_type": "PERSON_NODE"
      },
      "links": {
        "self": "/v1/entities/1000001"
      }
    },
    {
      "id": "1000002",
      "type": "entities",
      "attributes": {
        "currency_factor": "USD",
        "ownership_type": "PERCENT_BASED",
        "original_name": "X092849032",
        "display_name": "Citco",
        "model_type": "FINANCIAL_ACCOUNT",
        "is_rolled_up": false
      },
      "links": {
        "self": "/v1/entities/1000002"
      }
    },
    {
      "id": "1000003",
      "type": "entities",
      "attributes": {
        "currency_factor": "USD",
        "ownership_type": "PERCENT_BASED",
        "original_name": "Adam Irrevocable Trust",
        "model_type": "TRUST"
      },
      "links": {
        "self": "/v1/entities/1000003"
      }
    }
  ],
  "links": {
    "next": null
  }
}
```

Retrieve an Entity

```
GET /v1/entities/:entity-id
```

Retrieve all available attribute details for the specified entity.

Example

```
# Request
```

```
GET https://examplefirm.com/api/v1/entities/1000001

# Response
HTTP/1.1 200

{
  "data": {
    "id": "1000001",
    "type": "entities",
    "attributes": {
      "currency_factor": "USD",
      "original_name": "Adam Smith",
      "model_type": "PERSON_NODE"
    },
    "links": {
      "self": "/v1/entities/1000001"
    }
  }
}
```

Create an Entity

```
POST /v1/entities/
```

Add a new entity to your firm. Returns the full details of the new entity.

Required attributes: "original_name", "currency_factor", "model_type"

Example

```
# Request
POST https://examplefirm.com/api/v1/entities

{
  "data": {
    "type": "entities",
    "attributes": {
      "original_name": "New entity",
      "currency_factor": "USD",
      "model_type": "PERSON_NODE"
    }
  }
}

# Response
HTTP/1.1 201 Created

{
  "data": {
 "id": "1111",
    "type": "entities",
    "attributes": {
      "original_name": "New entity",
      "currency_factor": "USD",
      "model_type": "PERSON_NODE"
    },
    "links": {
      "self": "/v1/entities/1111"
    }
  }
}
```

Create Multiple Entities

```
POST /v1/entities/
```

Add multiple new entities to your firm. Returns the full details of the new entities.

Required attributes: "original_name", "currency_factor", "model_type"

Example

```
# Request
POST https://examplefirm.com/api/v1/entities

{
  "data": {
    "type": "entities",
    "attributes": {
      "original_name": "Adam Smith",
      "currency_factor": "USD",
      "model_type": "PERSON_NODE"
    },
    "type": "entities",
    "attributes": {
      "original_name": "Smith Family Trust",
      "currency_factor": "USD",
      "model_type": "TRUST"
    }
  }
}

# Response
HTTP/1.1 201 Created

{
  "data": {
 "id": "1111",
       "type": "entities",
       "attributes": {
          "original_name": "New entity",
          "currency_factor": "USD",
          "model_type": "PERSON_NODE"
    },
      "links": {
        "self": "/v1/entities/1111"
    },
        "id": "1112",
        "type": "entities",
        "attributes": {
            "original_name": "Smith Family Trust",
      "currency_factor": "USD",
            "model_type": "TRUST"
    },
      "links": {
        "self": "/v1/entities/1112"
    }
  }
}
```

Update an Entity

```
PATCH /v1/entities/:entity-id
```

Update standard attribute values for an existing client, holding company, or trust, or update custom attribute values for any existing entity. Returns the full details of the entity updated.

Note: There are no restrictions on updating custom attributes via the API.

Example

```
# Request
PATCH https://examplefirm.com/api/v1/entities/1111

{
  "data": {
 "id": "1111"
       "type": "entities",
       "attributes": {
         "original_name": "Updated entity"
    }
  }
}

# Response
HTTP/1.1 200

{
  "data": {
 "id": "1111",
       "type": "entities",
       "attributes": {
           "original_name": "Updated entity",
     "currency_factor": "USD",
           "model_type": "PERSON_NODE"
    },
        "links": {
           "self": "/v1/entities/1111"
    }
  }
}
```

Update Multiple Entities

```
PATCH /v1/entities
```

Updates standard attribute values for multiple existing client, trust, holding company, or financial account entities, or updates custom attribute values for any existing entity. Returns the full details of the updated entities.

Note: There are no restrictions on updating custom attributes via the API.

Example

In this example, the two entity names are being updated from “Adam Smith” and “Smith Family Trust,” as seen in the POST example above, to “Adam T. Smith” and “The Smith Family Trust,” respectively.

```
# Request
PATCH https://examplefirm.com/api/v1/entities

{
  "data": {
 "id": "1111"
       "type": "entities",
       "attributes": {
          "original_name": "Adam T. Smith"
    },
 "id": "1112"
```

```
        "type": "entities",
        "attributes": {
           "original_name": "The Smith Family Trust"
     }
   }
}

# Response
HTTP/1.1 200

{
  "data": {
 "id": "1111",
        "type": "entities",
        "attributes": {
           "original_name": "Adam T. Smith",
    "currency_factor": "USD",
           "model_type": "PERSON_NODE"
     },
       "links": {
         "self": "/v1/entities/1111"
     },
         "id": "1112",
         "type": "entities",
         "attributes": {
             "original_name": "The Smith Family Trust",
       "currency_factor": "USD",
             "model_type": "PERSON_NODE"
     },
        "links": {
           "self": "/v1/entities/1112"
     }
   }
}
```

Delete an Entity

```
DELETE /v1/entities/:entity-id
```

Delete an existing entity from your firm's data. Returns an empty response payload.

Restrictions: An entity cannot be deleted if it holds other entities. For example, you cannot delete a holding company if it owns a security.

Example

```
# Request
DELETE https://examplefirm.com/api/v1/entities/1111

# Response
HTTP/1.1 204 No Content
```

Delete Multiple Entities

```
DELETE /v1/entities
```

Delete multiple existing entities from your firm's data. Returns an empty response payload.

Restrictions: An entity cannot be deleted if it holds other entities. For example, you cannot delete a holding company if it owns a security.

Example

```
# Request
DELETE https://examplefirm.com/api/v1/entities

{
  "data": [ {
 "id": 1111,
       "type": "entities"
   },
      {
      "id": 1112,
      "type": "entities"
   } ]
}

# Response
HTTP/1.1 204 No Content
```

Groups API

The Groups API resource can be used to create, read, update, and delete groups stored in Addepar, as well the entities owned by those groups.

Each group is a set of client portfolios and their underlying assets. Using the API, you can review and revise details for one or more groups, add or remove group members, or delete groups.

Summary

Base Route	/v1/groups
Methods	GET, POST, PATCH, DELETE
Output Format	*JSON*
Pagination	Yes
Permissions Required	Create, edit, and delete groups

Attributes

The Groups resource supports the API attributes listed below.

Name	Description
name	Name. String. Required Example: "Adam Smith Clients"
[security attribute]	All *security attributes* can be applied to groups, unless otherwise noted.

Relationships

The Groups resource supports the API relationships listed below.

Name	Description	Editable
members	A list of all the entities that are associated to the group.	Yes

Restrictions

- The API user must have permission to "Create, edit, and delete groups".
- Group members must have one of the following model types.

Name	Model Type
Client	PERSON_NODE
Managed Fund	MANAGED_PARTNERSHIP
Trust	TRUST
Holding Company	HOLDING_COMPANY
Holding Account	FINANCIAL_ACCOUNT

Retrieve All Groups

```
GET /v1/groups
```

Returns a list of all groups the authenticated user has permission to access, details about when each was created, and the clients and entities, along with the relevant *security attributes*, it holds.

```
# Request
GET https://examplefirm.com/api/v1/groups

# Response
HTTP/1.1 200

{
  "data": [
    {
      "id": "1234",
      "type": "groups",
      "attributes": {
        "name": "Euler Family"
      },
      "relationships": {
        "members": {
          "links": {
            "self": "/v1/groups/1234/relationships/members",
            "related": "/v1/groups/1234/members"
          },
          "data": [
            {
              "type": "entities",
              "id": "1000001"
            },
            {
              "type": "entities",
              "id": "1000002"
            },
            {
              "type": "entities",
              "id": "1000003"
            }
          ]
        }
      },
      "links": {
        "self": "/v1/groups/1234"
      }
    },
```

```
    {
      "id": "5678",
      "type": "groups",
      "attributes": {
        "name": "Adam Smith Clients"
      },
      "relationships": {
        "members": {
          "links": {
            "self": "/v1/groups/5678/relationships/members",
            "related": "/v1/groups/5678/members"
          },
          "data": [
            {
              "type": "entities",
              "id": "2000001"
            },
            {
              "type": "entities",
              "id": "2000002"
            }
          ]
        }
      },
      "links": {
        "self": "/v1/groups/5678"
      }
    }
  ],
  "links": {
    "next": null
  }
}
```

Retrieve a Group

```
GET /v1/groups/:group-id
```

Returns details for a specified group, including details about when it was created and the client and entities, along with the relevant *security attributes*, it holds.

Example

```
# Request
GET https://examplefirm.com/api/v1/groups/5678

# Response
HTTP/1.1 200

{
  "data": {
    "id": "5678",
    "type": "groups",
    "attributes": {
      "name": "Adam Smith Clients"
    },
    "relationships": {
      "members": {
        "links": {
          "self": "/v1/groups/5678/relationships/members",
          "related": "/v1/groups/5678/members"
        },
```

```
        "data": [
          {
            "type": "entities",
            "id": "2000001"
          },
          {
            "type": "entities",
            "id": "2000002"
          }
        ]
      }
    },
    "links": {
      "self": "/v1/groups/5678"
    }
  }
}
```

Create a Group

```
POST /v1/groups/
```

Creates a new group. Returns the details of the group created.

Example

```
# Request
POST https://examplefirm.com/api/v1/groups

{
  "data": {
    "type": "groups",
    "attributes": {
      "name": "New Group"
    },
    "relationships": {
      "members": {
        "data": [
           {
            "type": "entities",
            "id": "2000001"
           },
           {
            "type": "entities",
            "id": "2000001"
           }
        ]
      }
    }
  }
}

# Response
HTTP/1.1 201 Created

{
  "data": {
    "id": 1111,
    "type": "groups",
    "attributes": {
      "name": "New Group"
    },
```

```
    "relationships": {
      "members": {
        "links": {
          "self": "/v1/groups/1111/relationships/members",
          "related": "/v1/groups/1111/members"
        },
        "data": [
           {
            "type": "entities",
            "id": "2000001"
           },
           {
            "type": "entities",
            "id": "2000001"
           }
        ]
      }
    },
    "links": {
      "self": "/v1/groups/1111"
    }
  }
}
```

Create Multiple Groups

```
POST /v1/groups/
```

Creates new groups. You you can specify group members and attributes like a group name and all *security attributes*, unless otherwise noted. Returns the details of each group created.

Example

```
# Request
POST https://examplefirm.com/api/v1/groups

{
  "data": [
      {
      "type": "groups",
      "attributes": {
        "name": "New Group 1"
      },
      "relationships": {
       "members": {
  "data": [
              {
              "type": "entities",
              "id": "2000001"
              },
              {
              "type": "entities",
              "id": "2000002"
              }
          ]
      }
  },
{
      "type": "groups",
      "attributes": {
        "name": "New Group 2"
        "sector": [
```

```
        {
           "date": "2017-01-01",
           "value": "Large Cap",
           "weight": 0.5
        }
     ]
   },
      "relationships": {
         "members": {
  "data": [
              {
              "type": "entities",
              "id": "2000003"
              },
              {
              "type": "entities",
              "id": "2000004"
              }
           ]
         }
       }
     }
  ]
}


# Response
HTTP/1.1 201 Created

{
  "data": [
        {
  "id": "1113"
        "type": "groups",
        "attributes": {
          "name": "New Group 1"
         },
        "relationships": {
          "members": {
            "links": {
              "self": "/v1/groups/1113/relationships/members",
              "related": "/v1/groups/1113/members"
            },
             "data": [
                {
                "type": "entities",
                "id": "2000001"
                },
                {
                "type": "entities",
                "id": "2000002"
                }
            ]
        }
    },
    "links": {
      "self": "/v1/groups/1113"
    }
  },
{
 "id": "1114"
       "type": "groups",
       "attributes": {
          "name": "New Group 2"
```

```
          "sector": [
             {
             "date": "2017-01-01",
             "value": "Large Cap",
             "weight": 0.5
              }
           ]
        },
        "relationships": {
            "members": {
              "links": {
                  "self": "/v1/groups/1114/relationships/members",
                  "related": "/v1/groups/1114/members"
               },
              "data": [
                {
                "type": "entities",
                "id": "2000003"
                },
               {
                "type": "entities",
                "id": "2000004"
                }
            ]
        }
    },
    "links": {
      "self": "/v1/groups/1114"
    }
  }
 ]
}
```

Update a Group

```
PATCH /v1/groups/:group-id
```

Updates the group name. Returns the updated details, including the name and all attributes.

Example

In this example, the group name is being updated from “Adam Smith Clients,” as seen in the POST example above, to “Updated Group Name.”

```
# Request
PATCH https://examplefirm.com/api/v1/groups/1111

{
  "data": {
 "id": "5678"
       "type": "groups",
       "attributes": {
         "name": "Updated Group Name"
    }
  }
}

# Response
HTTP/1.1 200

{
  "data": {
```

```
 "id": "5678"
       "type": "groups",
       "attributes": {
          "name": "Updated Group Name"
    },
    "relationships": {
      "members": {
        "links": {
          "self": "/v1/groups/5678/relationships/members",
          "related": "/v1/groups/5678/members"
        },
        "data": [
        {
            "type": "entities",
            "id": "2000001"
             },
             {
            "type": "entities",
            "id": "2000002"
          }
      ]
      }
    },
    "links": {
      "self": "/v1/groups/5678"
    }
  }
}
```

Delete a Group

```
DELETE /v1/groups/:group-id
```

Deletes a group from the firm.

Example

```
# Request
DELETE https://examplefirm.com/api/v1/groups/1111

# Response
HTTP/1.1 204 No Content
```

Delete Multiple Groups

```
DELETE /v1/groups
```

Deletes multiple existing groups from your firm’s data. Returns an empty response payload.

Example

```
# Request
DELETE https://examplefirm.com/api/v1/groups

{
  "data": [ {
 "id": "1111",
       "type": "groups"
   },
      {
      "id": "1112",
```

```
      "type": "groups"
    }
  ]
}

# Response
HTTP/1.1 204 No Content
```

Get all Group Member Relationships

```
GET /v1/groups/:group-id/relationships/members
```

Returns a list of all relationships for the members of the specified group.

Example

```
# Request
GET https://examplefirm.com/api/v1/groups/5678/relationships/members

# Response
HTTP/1.1 200

{
  "data": [
    {
       "type": "entities",
       "id": "2000001"
     },
     {
       "type": "entities",
       "id": "2000002"
     }
  ]
}
```

Add New Member Relationships to a Group

```
POST /v1/groups/:group-id/relationships/members
```

Adds existing entities as members to a group.

Example

```
# Request
POST https://examplefirm.com/api/v1/groups/5678/relationships/members

{
  "data": [
    {
      "type": "entities",
      "id": "100"
    },
    {
      "type": "entities",
      "id": "101"
    }
  ]
}

# Response
```

```
HTTP/1.1 204 No Content
```

Replace Existing Group Members

```
PATCH /v1/groups/:group-id/relationships/members
```

Replaces all the existing group members with new ones.

Example

```
# Request
PATCH https://examplefirm.com/api/v1/groups/5678/relationships/members

{
  "data": [
    {
      "type": "entities",
      "id": "100"
    },
    {
      "type": "entities",
      "id": "101"
    }
  ]
}

# Request
HTTP/1.1 204 No Content
```

Remove Group Members

```
DELETE /v1/groups/:group-id/relationships/members
```

Remove the specified entities as members of the group.

Example

```
# Request
DELETE https://examplefirm.com/api/v1/groups/5678/relationships/members

{
  "data": [
    {
      "type": "entities",
      "id": "2000001"
    }
  ]
}

# Response
HTTP/1.1 204 No Content
```

Retrieve all Group Members

```
GET /v1/groups/:group-id/members
```

Returns details of all the entities that are members of the specified group. Refer to the *entities API resource documentation* for information about the supported attributes and relationships, including sample responses.

Example

```
# Request
GET https://examplefirm.com/api/v1/groups/5678/members

# Response
HTTP/1.1 200

{
  "data": [
    {
      "id": "2000001",
      "type": "entities",
      "attributes": {
        "currency_factor": "USD",
        "original_name": "Adam Smith",
        "model_type": "PERSON_NODE"
      },
      "links": {
        "self": "/v1/entities/2000001"
      }
    },
    {
      "id": "2000002",
      "type": "entities",
      "attributes": {
        "currency_factor": "USD",
        "ownership_type": "PERCENT_BASED",
        "original_name": "X092849032",
        "display_name": "Citco",
        "model_type": "FINANCIAL_ACCOUNT",
        "is_rolled_up": false
      },
      "links": {
        "self": "/v1/entities/2000002"
      }
    }
  ],
  "links": {
    "next": null
  }
}
```

Files API

The Files API resource can be used to create, read, update, and delete files stored in Addepar, as well the portfolios, groups, and assets associated with those files.

Using the API, you can upload and download files (including both files currently available in Addepar and those that have been deleted), access file metadata (for example, details about when a file was created or deleted), and review and revise how files are organized and shared amongst entities and groups in Addepar.

Summary

Base Route	/v1/files
Methods	GET, POST, PATCH, DELETE
Output Format	*JSON*
Pagination	Yes

Permissions Required	View: Files View-only permission Edit: Files Full permission to edit

Attributes

The Files resource supports the API attributes listed below.

Name	Output	Editable	Required
name	**Example:** "Sample File.txt" String including the file's name	Yes	Yes
content_type	**Example:**"application/pdf" or "application/txt" String including the file's content type	No	No
created_at	**Example:**"2014-12-01T13:22:40Z" DateTime value for the file creation	No	No
deleted_at	**Example:**"2014-12-01T13:22:40Z" DateTime value for the file deletion	No	No
bytes	**Example:**"100000" Long integer value representing the size of the file, in bytes	No	No

Relationships

The Files resource supports the API relationships listed below.

Name	Description	Editable
associated_groups	A list of all the groups that are associated to the file.	Yes
associated_entities	A list of all the entities that are associated to the file.	Yes

Filter Parameters

The Get All Files (/v1/files), Get a Single File (/v1/files/:file-id), and Get All Archived Files (/v1/archive/files) resources support the API filter parameters listed below.

Filter	Description	Example
filter[files][created_after]	Return a filtered list of files created after a specific time.	/v1/files/?filter[files][created_after]=2017-04-06T20:12:45Z
filter[files][created_before]	Return a filtered list of files created before a specific time.	/v1/files/?filter[files][created_before]=2017-04-06T20:12:45Z

Filter	Description	Example
filter[files][created_after] &filter[files][created_before]	Both filters can be used simultaneously to return a filtered list of files created within a specific time range.	/v1/files/?filter[files] [created_after]=2017-04-05T20:12:45Z&filter[files] [created_before]=2017-04-06T20:12:45Z
filter[files][entityId]	Returns a filtered list of files associated with the specified entity.	/v1/files?[files][entityId]=100
filter[files][groupId]	Returns a filtered list of files associated with the specified group.	/v1/files?[files][groupId]=20
filter[files][entityId]&filter[files] [groupId]	If IDs for both an entity and a group are included in a request, the API ignores the group and returns only the files associated with the entity.	/v1/files?[files] [entityId]=100&filter[files] [groupId]=20

Note: Filters for the Files API require the ISO 8601 date format.

Restrictions

- The API user must have permission to view files in Addepar.
- The files and relationships accessible via the API will be restricted to the portfolios the user has permission to access.

Retrieve All Files

```
GET /v1/files
```

Returns a list of all files the authenticated user has permission to access, as well as details about when each was created, the type of content it holds, its name and size, and the groups and entities associated with it.

Example

```
# Request
GET https://examplefirm.com/api/v1/files

# Response
HTTP/1.1 200

{
  "data": [
    {
      "id": "123",
      "type": "files",
      "attributes": {
        "content_type": "application/pdf",
        "bytes": 256201,
        "name": "Sample.pdf",
        "created_at": "2014-06-20T20:55:07Z"
      },
      "relationships": {
        "associated_groups": {
          "links": {
            "self": "/v1/files/123/relationships/associated_groups",
            "related": "/v1/files/123/associated_groups"
          },
          "data": [
            {
              "type": "groups",
              "id": "1234"
            },
```

```
            {
              "type": "groups",
              "id": "5678"
            }
          ]
        },
        "associated_entities": {
          "links": {
            "self": "/v1/files/123/relationships/associated_entities",
            "related": "/v1/files/123/associated_entities"
          },
          "data": [
            {
              "type": "entities",
              "id": "10000"
            },
            {
              "type": "entities",
              "id": "10001"
            }
          ]
        }
      },
      "links": {
        "self": "/v1/files/123"
      }
    },
    {
      "id": "345",
      "type": "files",
      "attributes": {
        "content_type": "application/pdf",
        "bytes": 798121,
        "name": "Report.pdf",
        "created_at": "2014-06-20T20:55:19Z"
      },
      "relationships": {
        "associated_groups": {
          "links": {
            "self": "/v1/files/345/relationships/associated_groups",
            "related": "/v1/files/345/associated_groups"
          },
          "data": []
        },
        "associated_entities": {
          "links": {
            "self": "/v1/files/345/relationships/associated_entities",
            "related": "/v1/files/345/associated_entities"
          },
          "data": [
            {
              "type": "entities",
              "id": "123400"
            }
          ]
        }
      },
      "links": {
        "self": "/v1/files/345"
      }
    }
  ],
  "links": {
    "next": null
```

```
  }
}
```

Retrieve a File

```
GET /v1/files/:file-id
```

Returns the details of the specified file including information about when it was created, the type of content it holds, its name and size, and the groups and entities associated with it.

Example

```
# Request
GET https://examplefirm.com/api/v1/files/123

# Response
HTTP/1.1 200

{
  "data": {
    "id": "123",
    "type": "files",
    "attributes": {
      "content_type": "application/pdf",
      "bytes": 256201,
      "name": "Sample.pdf",
      "created_at": "2014-06-20T20:55:07Z"
    },
    "relationships": {
      "associated_groups": {
        "links": {
          "self": "/v1/files/123/relationships/associated_groups",
          "related": "/v1/files/123/associated_groups"
        },
        "data": [
          {
            "type": "groups",
            "id": "1234"
          },
          {
            "type": "groups",
            "id": "5678"
          }
        ]
      },
      "associated_entities": {
        "links": {
          "self": "/v1/files/123/relationships/associated_entities",
          "related": "/v1/files/123/associated_entities"
        },
        "data": [
          {
            "type": "entities",
            "id": "10000"
          },
          {
            "type": "entities",
            "id": "10001"
          }
        ]
      }
    },
    "links": {
```

```
      "self": "/v1/files/123"
    }
  }
}
```

Download a File

```
GET /v1/files/:file-id/download
```

Returns the raw contents of a specific file.

Example

```
# Request
GET https://examplefirm.com/api/v1/files/123/download

# Response
HTTP/1.1 200
Content-Disposition: attachment; filename="Sample.pdf"
Content-Type: application/binary

<RAW_FILE_DATA>
```

Create a File

```
POST /v1/files/
```

Adds a new file to your firm. Metadata is not required, but can be specified in order to change the name of the file to a new name in Addepar, or to link the file to entities and groups.

Important: The file extension provided in the metadata must match the file extension of filename in "Content-Disposition".

Example

```
# Request
POST https://examplefirm.com/api/v1/files
Content-Type: multipart/form-data; boundary=<UNIQUE_BOUNDARY>
Content-Length: 2198

--<UNIQUE_BOUNDARY>
Content-Disposition: form-data; name="file"; filename="Sample.txt"
Content-Type: text/plain

<RAW_FILE_DATA>

--<UNIQUE_BOUNDARY>
Content-Disposition: form-data; name="metadata"

{
  "data": {
    "type": "files",
    "attributes": {
      "name": "Sample.txt",
    }
  }
}
--<UNIQUE_BOUNDARY>--

# Response
HTTP/1.1 201 Created

{
```

```
  "data": {
 "id": 1111,
    "type": "files",
    "attributes": {
      "content_type": "application/pdf",
      "bytes": 256201,
      "name": "Sample.txt",
      "created_at": "2017-01-01T20:55:07Z"
    },
    "relationships": {
      "associated_groups": {
        "links": {
          "self": "/v1/files/1111/relationships/associated_groups",
          "related": "/v1/files/111113/associated_groups"
        },
        "data": []
      },
      "associated_entities": {
        "links": {
          "self": "/v1/files/1111/relationships/associated_entities",
          "related": "/v1/files/1111/associated_entities"
        },
        "data": []
      }
    },
    "links": {
      "self": "/v1/files/1111"
    }
  }
}
```

Update a File

```
PATCH /v1/files/:file-id
```

Updates file attributes and relationships.

Example

```
# Request
PATCH https://examplefirm.com/api/v1/files/1111

{
  "data": {
 "id": 1111,
    "type": "files",
    "attributes": {
      "name": "RenamedFile.txt"
    }
}

# Response
HTTP/1.1 200

{
  "data": {
 "id": 1111,
    "type": "files",
    "attributes": {
      "content_type": "application/pdf",
      "bytes": 256201,
      "name": "RenamedFile.txt",
      "created_at": "2017-01-01T20:55:07Z"
```

```
    },
    "relationships": {
      "associated_groups": {
        "links": {
          "self": "/v1/files/1111/relationships/associated_groups",
          "related": "/v1/files/111113/associated_groups"
        },
        "data": []
      },
      "associated_entities": {
        "links": {
          "self": "/v1/files/1111/relationships/associated_entities",
          "related": "/v1/files/1111/associated_entities"
        },
        "data": []
      }
    },
    "links": {
      "self": "/v1/files/1111"
    }
  }
}
```

Delete a File

```
DELETE /v1/files/:file-id
```

Archives the file within your firm instance. The archived file and its associated metadata will remain available through separate routes (described below).

Example

```
# Request
DELETE https://examplefirm.com/api/v1/files/1111

# Response
HTTP/1.1 204 No Content
```

Retrieve All Groups Associated with a File

```
GET /v1/files/:file-id/relationships/associated_groups
```

Returns a list of groups with which the file is associated.

Example

```
# Request
GET https://examplefirm.com/api/v1/files/123/relationships/associated_groups

# Response
HTTP/1.1 200
{
  "links": {
    "self": "/v1/files/123/relationships/associated_groups",
    "related": "/v1/files/123/associated_groups"
  },
  "data": [
    {
      "type": "groups",
      "id": "1234"
    },
    {
```

```
      "type": "groups",
      "id": "5678"
    }
  ]
}
```

Retrieve a File's Associated Entity Relationships

```
GET /v1/files/:file-id/relationships/associated_entities
```

Returns a list of all the entities with which the file is associated.

Example

```
# Request
GET https://examplefirm.com/api/v1/files/123/relationships/
associated_entities

# Response
HTTP/1.1 200
{
  "links": {
    "self": "/v1/files/123/relationships/associated_entities",
    "related": "/v1/files/123/associated_entities"
  },
  "data": [
    {
      "type": "entities",
      "id": "10000"
    },
    {
      "type": "entities",
      "id": "10001"
    }
  ]
}
```

Associate a File with One or More Groups

```
POST /v1/files/:file-id/relationships/associated_groups
```

Creates new associations between the file and existing groups.

Example

```
# Request
POST https://examplefirm.com/api/v1/files/123/relationships/
associated_groups

{
  "data": [
    { "type": "groups", "id": "200" },
    { "type": "groups", "id": "201" }
  ]
}

# Response
HTTP/1.1 204 No Content
```

Associate a File with One or More Entities

```
POST /v1/files/:file-id/relationships/associated_entities
```

Creates new associations between the file and existing entities.

Example

```
# Request
POST https://examplefirm.com/api/v1/files/123/relationships/
associated_entities

{
  "data": [
    { "type": "entities", "id": "100" },
    { "type": "entities", "id": "101" }
  ]
}

# Response
HTTP/1.1 204 No Content
```

Replace Existing Group Associations

```
PATCH /v1/files/:file-id/relationships/associated_groups
```

Replaces all the currently associated groups with new group associations.

Example

```
# Request
PATCH https://examplefirm.com/api/v1/files/123/relationships/
associated_groups


{
  "data": [
    { "type": "groups", "id": "200" },
    { "type": "groups", "id": "201" }
  ]
}

# Response
HTTP/1.1 204 No Content
```

Replace Existing Entity Associations

```
PATCH /v1/files/:file-id/relationships/associated_entities
```

Replaces all the currently associated entities with new entity associations.

Example

```
# Request
PATCH https://examplefirm.com/api/v1/files/123/relationships/
associated_entities

{
  "data": [
    { "type": "entities", "id": "100" },
    { "type": "entities", "id": "101" }
  ]
}

# Response
HTTP/1.1 204 No Content
```

Remove Group Associations from a File

```
DELETE /v1/files/:file-id/relationships/associated_groups
```

Removes the specified group associations from the file.

Example

```
# Request
DELETE https://examplefirm.com/api/v1/files/123/relationships/
associated_groups

{
  "data": [
    { "type": "groups", "id": "5678" }
  ]
}

# Response
HTTP/1.1 204 No Content
```

Remove Entity Associations from a File

```
DELETE /v1/files/:file-id/relationships/associated_entities
```

Removes the specified entity associations from the file.

Example

```
# Request
DELETE https://examplefirm.com/api/v1/files/123/relationships/
associated_entities

{
  "data": [
    { "type": "entities", "id": "10000" }
  ]
}

# Response
HTTP/1.1 204 No Content
```

Retrieve all Groups Associated with a File

```
GET /v1/files/:file-id/associated_groups
```

Returns the details of all the groups associated with the specified file. Refer to *groups resource API documentation* for information about the supported attributes and relationships, as well as sample responses.

Retrieve all Entities Associated with a File

```
GET /v1/files/:file-id/associated_entities
```

Returns the details of all the entities associated with the specified file. Refer to *entities resource API documentation* for information about the supported attributes and relationships, as well as sample responses.

Retrieve all Archived Files

```
GET /v1/archive/files
```

Returns a list of all archived (deleted) files, as well as information about when each was created, the type of content it holds, its name and size, the groups and entities associated with it, and when it was deleted.

Example

```
# Request
GET https://examplefirm.com/api/v1/archive/files

# Response
HTTP/1.1 200

{
  "data": [
    {
      "id": "319",
      "type": "files",
      "attributes": {
        "content_type": "text/plain",
        "bytes": 21607,
        "name": "archived_transactions.csv",
        "created_at": "2017-03-15T20:31:49Z",
        "deleted_at": "2017-03-15T20:32:16Z"
      },
      "relationships": {
        "associated_groups": {
          "links": {
            "self": "/v1/archive/files/319/relationships/associated_groups",
            "related": "/v1/archive/files/319/associated_groups"
          },
          "data": []
        },
        "associated_entities": {
          "links": {
            "self": "/v1/archive/files/319/relationships/
associated_entities",
            "related": "/v1/archive/files/319/associated_entities"
          },
          "data": [
            {
              "type": "entities",
              "id": "22"
            }
          ]
        }
      },
      "links": {
        "self": "/v1/archive/files/319"
      }
    }
  ],
  "links": {
    "next": null
  }
}
```

Retrieve an Archived File

```
GET /v1/archive/files/:file-id
```

Returns an archived (deleted) file, as well as information about when it was created, the type of content it holds, its name and size, the groups and entities associated with it, and when it was deleted.

Example

```
# Request
GET https://examplefirm.com/api/v1/archive/files/319
```

```
# Response
HTTP/1.1 200

{
  "data": {
    "id": "319",
    "type": "files",
    "attributes": {
      "content_type": "text/plain",
      "bytes": 21607,
      "name": "archived_transactions.csv",
      "created_at": "2017-03-15T20:31:49Z",
      "deleted_at": "2017-03-15T20:32:16Z"
    },
    "relationships": {
      "associated_groups": {
        "links": {
          "self": "/v1/archive/files/319/relationships/associated_groups",
          "related": "/v1/archive/files/319/associated_groups"
        },
        "data": []
      },
      "associated_entities": {
        "links": {
          "self": "/v1/archive/files/319/relationships/associated_entities",
          "related": "/v1/archive/files/319/associated_entities"
        },
        "data": [
          {
            "type": "entities",
            "id": "22"
          }
        ]
      }
    },
    "links": {
      "self": "/v1/archive/files/319"
    }
  }
}
```

Download an Archived File

```
GET /v1/archive/files/:file-id/download
```

Returns the raw contents of an archived file.

Example

```
# Request
GET https://examplefirm.com/api/v1/archive/files/319/download

# Response
HTTP/1.1 200
Content-Disposition: attachment; filename="Sample.pdf"
Content-Type: application/binary

<RAW_FILE_DATA>
```

Users API

The Users API resource can be used to read Addepar user data, including information about the user's credentials and the data each user has permission to access.

Users in Addepar represent access credentials and associated permissions to the Addepar's advisor portal within a given firm.

Summary

Base Route	/v1/users
Methods	GET
Output Format	*JSON*
Pagination	No
Permissions Required	Manage users and permissions

Attributes

The Users resource supports the API attributes listed below.

Name	Description
email	**Example:** "adam@smith.com" String containing the email address used for authentication
first_name	**Example:** "Adam" String containing the user's first name
last_name	**Example:** "Smith" String containing the user's last name
saml_user_id	**Example:** "asmith" String containing the SAML ID assigned to the user (only applicable if SSO is enabled)
all_data_access	**Example:** "false" Boolean flag indicating whether the user has permission to access all current and future portfolio data
two_factor_auth_enabled	**Example:** "false" Boolean flag indicating whether the user has 2-factor authentication enabled
external_user_id	**Example:** "A12345" Alphanumeric string containing the firm’s unique ID for the user (an employee ID number, a user ID from a human resources system, etc.)

Relationships

The Users resource supports the API relationships listed below.

Name	Description	Editable
permissioned_groups	A list of all the groups the user has permission to access.	No
permissioned_entities	A list of all the entities the user has permission to access.	No

Restrictions

- The Users resource can only be used to read user details: creating, updating, and deleting users is not supported via the API.
- User details are only provided for Addepar application users, and do not include users and credentials for the client portal.
- Tool permissions for users are not currently included in the response.

Retrieve All Users

```
GET /v1/users
```

Returns a list containing the details of each user, including the user's name, email, and access information.

Example

```
# Request
GET https://examplefirm.addepar.com/api/v1/users

# Response
HTTP/1.1 200

{
  "data": [
    {
      "id": "1000",
      "type": "users",
      "attributes": {
        "two_factor_auth_enabled": true,
        "all_data_access": true,
        "external_user_id": "A12345",
        "email": "user1@addepar.com",
        "first_name": "Adam",
        "last_name": "Smith"
      },
      "relationships": {
        "permissioned_entities": {
          "links": {
            "self": "/v1/users/1000/relationships/permissioned_entities",
            "related": "/v1/users/1000/permissioned_entities"
          },
          "data": []
        },
        "permissioned_groups": {
          "links": {
            "self": "/v1/users/1000/relationships/permissioned_groups",
            "related": "/v1/users/1000/permissioned_groups"
          },
          "data": []
        }
      },
      "links": {
        "self": "/v1/users/1000"
```

```
      }
    },
    {
      "id": "2000",
      "type": "users",
      "attributes": {
        "two_factor_auth_enabled": false,
        "all_data_access": false,
        "external_user_id": "A67890",
        "email": "user2@addepar.com",
        "first_name": "Jane",
        "last_name": "Doe"
      },
      "relationships": {
        "permissioned_entities": {
          "links": {
            "self": "/v1/users/2000/relationships/permissioned_entities",
            "related": "/v1/users/2000/permissioned_entities"
          },
          "data": [
            {
               "type": "entities",
               "id": 10000
            },
            {
               "type": "entities",
               "id": 10001
            }
          ]
        },
        "permissioned_groups": {
          "links": {
            "self": "/v1/users/2000/relationships/permissioned_groups",
            "related": "/v1/users/2000/permissioned_groups"
          },
          "data": [
            {
               "type": "entities",
               "id": 20000
            },
            {
               "type": "entities",
               "id": 20001
            }
          ]
        }
      },
      "links": {
        "self": "/v1/users/2000"
      }
    }
  ],
  "links": {
    "next": null
  }
}
```


Retrieve a User

```
GET /v1/users/:user-id
```

Returns details for a specified user, including the user's name, email, and access information.

Example

```
# Request
GET https://examplefirm.addepar.com/api/v1/users/2000

# Response
HTTP/1.1 200

{
  "id": "2000",
  "type": "users",
  "attributes": {
    "two_factor_auth_enabled": false,
    "all_data_access": false,
    "external_user_id": "A67890",
    "email": "user2@addepar.com",
    "first_name": "Jane",
    "last_name": "Doe"
  },
  "relationships": {
    "permissioned_entities": {
      "links": {
        "self": "/v1/users/2000/relationships/permissioned_entities",
        "related": "/v1/users/2000/permissioned_entities"
      },
      "data": [
        {
           "type": "entities",
           "id": 10000
        },
        {
           "type": "entities",
           "id": 10001
        }
      ]
    },
    "permissioned_groups": {
      "links": {
        "self": "/v1/users/2000/relationships/permissioned_groups",
        "related": "/v1/users/2000/permissioned_groups"
      },
      "data": [
        {
           "type": "entities",
           "id": 20000
        },
        {
           "type": "entities",
           "id": 20001
        }
      ]
    }
  },
  "links": {
    "self": "/v1/users/2000"
  }
}
```

Retrieve Users by External User ID

```
POST /v1/users/external_user_id
```

Returns a list containing the details of each user, including the user's name, email, and data access information.

Note: Currently you can only retrieve the external user ID for a user in the API. The external user ID is included in the response for each GET request. You can *add the external user ID* for each user directly in the application.

Example

```
# Request
GET https://examplefirm.addepar.com/api/v1/external_user_id

{
  "data": {
    "type": "external_user_id",
    "attributes": {
      "external_user_ids": [
        "A12345",
        "A67890"
      ]
    }
  }
}

# Response
HTTP/1.1 200

{
  "data": [
    {
      "id": "1000",
      "type": "users",
      "attributes": {
        "two_factor_auth_enabled": true,
        "all_data_access": true,
        "external_user_id": "A12345",
        "email": "user1@addepar.com",
        "first_name": "Adam",
        "last_name": "Smith"
      },
      "relationships": {
        "permissioned_entities": {
          "links": {
            "self": "/v1/users/1000/relationships/permissioned_entities",
            "related": "/v1/users/1000/permissioned_entities"
          },
          "data": []
        },
        "permissioned_groups": {
          "links": {
            "self": "/v1/users/1000/relationships/permissioned_groups",
            "related": "/v1/users/1000/permissioned_groups"
          },
          "data": []
        }
      },
      "links": {
        "self": "/v1/users/1000"
      }
    },
    {
      "id": "2000",
      "type": "users",
      "attributes": {
```

```
        "two_factor_auth_enabled": false,
        "all_data_access": false,
        "external_user_id": "A67890",
        "email": "user2@addepar.com",
        "first_name": "Jane",
        "last_name": "Doe"
      },
      "relationships": {
        "permissioned_entities": {
          "links": {
            "self": "/v1/users/2000/relationships/permissioned_entities",
            "related": "/v1/users/2000/permissioned_entities"
          },
          "data": [
            {
               "type": "entities",
               "id": 10000
            },
            {
               "type": "entities",
               "id": 10001
            }
          ]
        },
        "permissioned_groups": {
          "links": {
            "self": "/v1/users/2000/relationships/permissioned_groups",
            "related": "/v1/users/2000/permissioned_groups"
          },
          "data": [
            {
               "type": "entities",
               "id": 20000
            },
            {
               "type": "entities",
               "id": 20001
            }
          ]
        }
      },
      "links": {
        "self": "/v1/users/2000"
      }
    }
  ],
  "links": {
    "next": null
  }
}
```


Retrieve All Groups the User Can Access

```
GET /v1/users/:user-id/relationships/permissioned_groups
```

Returns a list of all the groups the specified user has permission to access. Refer to the *groups API resource* documentation for details about the supported attributes and relationships, including sample responses.

Example

```
# Request
```

```
GET https://examplefirm.addepar.com/api/v1/users/5678/relationships/
permissioned_groups

# Response
HTTP/1.1 200

{
 "data": [
    {
      "type": "groups",
      "id": "200"
    },
    {
      "type": "groups",
      "id": "201"
    }
  ]
}
```

Retrieve All Entities the User Can Access

```
GET /v1/users/:user-id/relationships/permissioned_entities
```

Returns a list of all the entities the specified user can access. Refer to the entities API resource documentation for details about the supported attributes and relationships, including sample responses.

Example

```
# Request
GET https://examplefirm.addepar.com/api/v1/users/5678/relationships/
permissioned_entities

# Response
HTTP/1.1 200

{
 "data": [
    {
      "type": "entities",
      "id": "10000"
    },
    {
      "type": "entities",
      "id": "10001"
    }
  ]
}
```

Jobs API

The asynchronous Jobs API allows you to reliably export large Analysis views and download the results any time within 24 hours of the initial request.

You can use the Jobs API to:

- Issue API requests that cannot be completed within the current, synchronous API timeout limits
- Avoid maintaining an open HTTP connection while waiting for a response from the API
- Protect long-running API requests from the effects of unreliable networks

For information about the Analysis views you can download, refer to the *Portfolio API* documentation.

Summary

Base Route	/v1/jobs
Methods	GET, POST
Output Format	*JSON* (job status) Multiple (download)
Pagination	No
Permissions Required	Multiple (job creation)

Attributes

The Jobs resource supports the API attributes listed below.

Name	Description
job_type	**Example:** "portfolio_view_results" String containing job type
parameters	JSON object containing parameters relevant to a specific job type.

Relationships

The Jobs resource supports the API relationships listed below.

Name	Description
creator	The user who created the job.

Restrictions

- Currently only job_type "portfolio_view_results" is supported.
- Relevant permissions pertaining to each job type are enforced at job creation.
- A user may not access jobs created by another user.

Job Statuses

Status	Meaning
Queued	Job has not yet started processing by Addepar servers
In Progress	Job is currently being processed by Addepar
Completed	Job has finished and the results can be retrieved from the url in the Location Header
Timed Out	Job results have expired and results are deleted after 24 hours since job creation.
Error	Job processing failed due to an error. Details are provided within the error field of the response.

Create a job

Visit the *Portfolio API* documentation for information about creating an analysis data job.

Check the Status of All Jobs

```
GET /v1/jobs
```

Returns the status and parameters for all current jobs.

Example

```
# Request
GET https://examplefirm.com/api/v1/jobs

# Response
HTTP/1.1 200

{
  "data": [
    {
      "id": "13",
      "type": "jobs",
      "attributes": {
        "job_type": "PORTFOLIO_VIEW_RESULTS",
        "parameters": {
          "view_id": 2,
          "portfolio_type": "entity",
          "portfolio_id": 22,
          "output_type": "tsv",
          "start_date": "2011-12-31",
          "end_date": "2013-01-15"
        },
        "status": "Completed"
      },
      "links": {
        "self": "/v1/jobs/13"
      }
    },
    {
      "id": "14",
      "type": "jobs",
      "attributes": {
        "job_type": "PORTFOLIO_VIEW_RESULTS",
        "parameters": {
          "view_id": 1,
          "portfolio_type": "entity",
          "portfolio_id": 10,
          "output_type": "csv",
          "start_date": "2016-01-01",
          "end_date": "2016-01-02"
        },
        "errors": [
          {
            "status": "400",
            "title": "Bad Request",
            "detail": "Invalid portfolio: ENTITY 10"
          }
        ],
        "status": "Error"
      },
      "links": {
        "self": "/v1/jobs/14"
      }
    }
  ],
  "links": {
    "next": null
  }
}
```

Check the Status of a Job

```
GET /v1/jobs/:job-id
```

Returns the status and parameters for a specific job.

Example

```
# Request
GET https://examplefirm.com/api/v1/jobs/13

# Response
HTTP/1.1 200

{
    "data": {
        "id": "13",
        "type": "jobs",
        "attributes": {
            "job_type": "PORTFOLIO_VIEW_RESULTS",
            "parameters": {
                "view_id": 2,
                "portfolio_type": "entity",
                "portfolio_id": 22,
                "output_type": "tsv",
                "start_date": "2011-12-31",
                "end_date": "2013-01-15"
            },
            "status": "Completed"
        },
        "relationships": {
            "creator": {
                "links": {
                    "self": "/v1/jobs/13/relationships/creator",
                    "related": "/v1/jobs/13/creator"
                },
                "data": {
                    "type": "users",
                    "id": "22"
                }
            }
        },
        "links": {
            "self": "/v1/jobs/13"
        }
    }
}
```

Download the Results of a Job

```
GET /v1/jobs/:job-id/download
```

Returns the raw contents of a specific file.

Example

```
# Request
GET https://examplefirm.com/api/v1/jobs/13/download

# Response
HTTP/1.1 200
Content-Disposition: attachment; filename="portfolio_data.tsv"
```

```
Content-Type: text/tsv

<DOWNLOAD_CONTENT>
```