Overview

The Quantitative Research Environment (QRE) API is an API for users to interact with QRE programmatically.

All APIs are hosted under https://api.factset.com. Authentication is handled using API Keys and authorization is handled using FactSet's in-house subscriptions product. You can find more information about using API Keys at https://developer.factset.com/authentication.

QRE API Endpoints

Calculation API

1.1 Calculations

POST /analytics/quant/qre/v1/calculations

This endpoint takes a python script and starts executing it within QRE

Input: python scripts within a json object

1.2 Get Calculation Status By ID

GET /analytics/quant/qre/v1/calculations/{id}

This is the endpoint to check on the progress of a calculation request.

A calculation execution can have a status of **"**failed", "pending", "completed"

1.3 Get calculation output for a specific calculation

GET /analytics/quant/qre/v1/calculations/{id}/output

This endpoint returns the specified output from the calculation

1.4 Get calculation log for a specific calculation

GET /analytics/quant/qre/v1/calculations/{id}/log

This endpoint returns the log from the calculation. If execution fails, you can see the error message from log endpoint

Upload API

2.1 Get Upload file

POST /analytics/quant/qre/v1/files/{server}/{file}

This is the endpoint to upload files to server

Possible server value: interactive, batch

You could upload files to both interactive and batch environment. “interactive” is the FactSet hosted JupyterLab research environment and “batch” environment is used for automation and production.

2.2 Get upload status by id

GET /analytics/quant/qre/v1/files/uploads/{id}

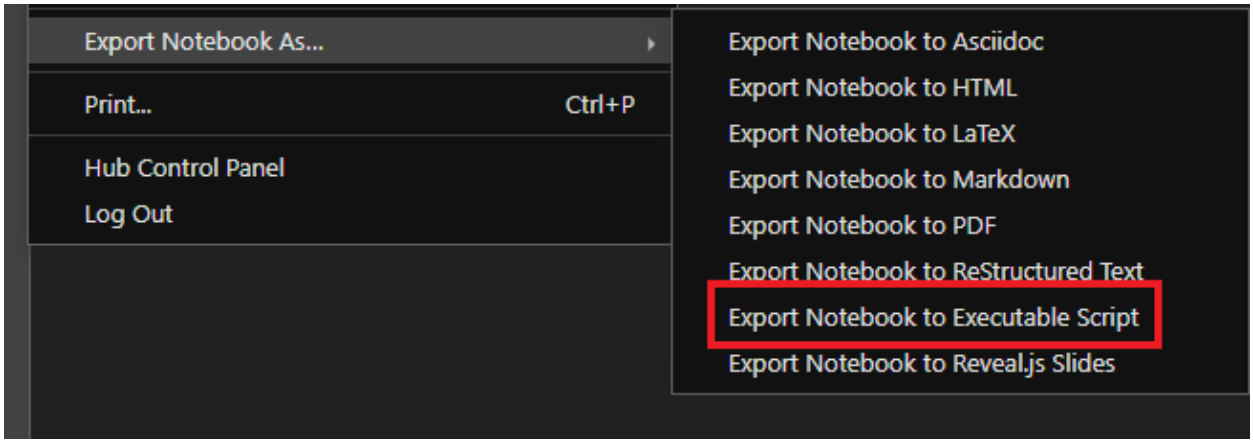
This is the endpoint to check on the progress of a previous upload request.

An upload execution can have a status of **“**failed”, “pending”, “completed”

Getting Your QRE Outputs

Below are some differences between running code in QRE and API:

- **Save your python script in a .py file**. You need to export your notebook as executable scripts to be used in QRE API. Go to File -> Export Notebook As -> Choose “Export Notebook to Executable Script”.



- **Output generations**. To generate outputs from QRE API, you need to use `openquant.output` module as part of python scripts. This module allows you to reformat the output into five different file formats. We currently support csv, parquet, feather, json and raw byte.

  Please refer to “Sample Python Script” snippet for details.

```
fds.quant.output.dataframe.to_json(df):
    '''Sets the script output to the passed DataFrame as JSON.
    Content-Type will be set to `application/json`.
    Parameters
    ----------
    df : DataFrame
        The DataFrame containing the values to output.
    '''

fds.quant.output.dataframe.to_csv(df):
    '''Sets the script output to the passed DataFrame as CSV.
    Content-Type will be set to `text/csv`.
    Parameters
    ----------
    df : DataFrame
        The DataFrame containing the values to output.
    '''

fds.quant.output.dataframe.to_feather(df):
   '''Sets the script output to the passed DataFrame as Feather.
    Content-Type will be set to `application/feather`.
    Parameters
    ----------
    df : DataFrame
        The DataFrame containing the values to output.
    '''

fds.quant.output.dataframe.to_parquet(df, compression='snappy'):
    '''Sets the script output to the passed DataFrame as Parquet.
    Content-Type will be set to `application/parquet`.
    Parameters
    ----------
    df : DataFrame
        The DataFrame containing the values to output.
    compression : {{'snappy', 'gzip', 'brotli', None}}, default 'snappy'
    Name of the compression to use. Use ``None`` for no compression.
    '''

fds.quant.output.raw.raw_bytes(data, content_type):
    '''Sets the script output to the passed bytes.
    Content-Type will be set to what is passed.
    Parameters
    ----------
    data : ByteString
        The raw bytes to output.
    content_type : String
        The content-type of the output.
    '''
```

Code Snippet

Sample Code to generate outputs

```
import pandas as pd
import numpy as np
import requests
import base64
import time

#username and api key
username = ''
api_key = ''
# Assemble basic auth user + pass
auth = bytes(f"{username.upper()}:{api_key}", "utf-8")
headers = {
    'Authorization': 'Basic %s' % str(base64.b64encode(auth).decode('ascii'))
}
api = 'https://api.factset.com/analytics/quant/qre'

# Load the script we want to run so we can send it to QRE
# Copy “sample python script” snippet to use here
script = ''
with open('./<your_python_script.py>') as f:
    script = f.read()

# start QRE calculation
r = requests.post(
    api + '/v1/calculations',
    headers=headers,
    json={'script': script} # Using `json` will encode the script as needed
)
print('HTTP Status: {}'.format(r.status_code))

# Calculation returns JSON body
print("Calculation_id:", r.json()['id'])
# Calculation also returns a `Location` header which contains a relative URL for
where to poll for calculation status
pollUrl = r.headers['Location']
# The unique id for this calculation. Used later to get the log and ouput
calculation_id = r.json()['id']
print("Polling url:", pollUrl)

# Poll for script completion
while True:
    #print("Polling for results...")
    r = requests.get(api + pollUrl, headers=headers)
    # Poll returns the same JSON body as the initial calculation but with an updated
status
    print("Polling result:", r.json()['status'])
    # 200 means the calculation is done
    # 202 means the calculation is still going
    # Anything > 299 means some service error occured
    if r.status_code == 200:
       break
```

```
    elif r.status_code > 299:
        print('Pickup request failed: ' + r.status_code)
        raise SystemExit('Pickup request failed')
    time.sleep(10)
print("Run finished" )

# Get script log
logs = requests.get(
    api + '/v1/calculations/'+calculation_id+'/log',
    headers=headers
)
print("Logs:" + logs.content.decode('utf8'))

# Get output (if there is any)
output = requests.get(
    api + '/v1/calculations/'+calculation_id+'/output',
    headers=headers
)

# Content-Type will be whatever you set it to be in the script
print("Output type:", output.headers.get('Content-Type'))

# convert output to dataframe
df = pd.read_csv(io.StringIO(output.content.decode('utf8')))
```

Sample Python Script

```
from fds.quant.screening import Screen
from fds.quant.universe import UnivLimit,
IdentifierUniverse,ScreeningExpressionUniverse,ScreeningDocumentUniverse
from fds.quant.dates import TimeSeries, RelativeDate,Frequency
from fds.quant.output.dataframe import to_csv

dates = TimeSeries(start=RelativeDate.ONE_CALENDAR_YEAR_AGO,
stop=RelativeDate.PREV_CLOSE, freq=Frequency.MONTHLY)

univ = ScreeningExpressionUniverse(expression=UnivLimit.SP500,time_series=dates)

screen = Screen(universe=univ, formulas=['P_PRICE(0)'],
columns=['price'],entire_universe=True)

screen.calculate()

df = screen.data

to_csv(df)
```

Sample Code to upload files

```
import requests
import base64
import time

#username and api key
username = ''
api_key = ''
# Assemble basic auth user + pass
auth = bytes(f"{username.upper()}:{api_key}", "utf-8")
headers = {
    'Authorization': 'Basic %s' % str(base64.b64encode(auth).decode('ascii'))
}
api = 'https://api.factset.com/analytics/quant/qre'

# Uploading a file
# # Load the file
fileToUpload = ''
with open('./<your upload files> ') as f:
    fileToUpload = f.read()

#uploaded filename
filename = <your upload filename>

#set environment to upload files
# - interactive
# - batch
env = 'interactive'
#env = 'batch'

#set uploadUrl
uploadUrl = api + '/v1/files/' + env + '/' + filename

r = requests.post(
    uploadUrl,
    headers=headers,
    data=fileToUpload)

upload_id = r.json()['id']
print("Upload id: " + upload_id)
pollUrl = r.headers['Location']

# Poll to see when the file is done uploading
while True:
    print("Polling for upload finish...")
    r = requests.get(api + pollUrl, headers=headers)
    if r.status_code == 200:
        break
    elif r.status_code > 299:
        print('Pickup request failed: ' + r.status_code)
        raise SystemExit('Pickup request failed')
    time.sleep(5)

print('Status: ' + r.json()['status'])
```