FACTSET 〉 SEE THE ADVANTAGE

ENGINES API - VERSION 3.0
Python Migration Guide
June 2021

Contents

1. Why v3?

The Engines API houses many of the calculation-based APIs in the Analytics API program, including PA, SPAR, Vault, Publisher, Fixed Income, and Optimizer. Version 2 was released in August 2019 and combined the previously distinct 'Batch API' with the single request v1 Engine APIs.

Engines v3 aims to improve both the developer experience and engine calculation efficiency with a light rework of engines endpoints, focusing on increased user control of the calculation flow. The Analytics API program designed v3 with the following overarching goals in mind:

1. Create more 'RESTful' API endpoints by treating submitted calculation parameters as a user resource that allow for CRUD operations (Create, Read, Update, Delete).
2. Improve calculation workflow efficiency (i.e. fewer redundant calculations)
3. Align with various FactSet-wide design standards

2. Summary of Changes

You can expect the following changes in v3 of the Engines APIs:

1. Support for cache-control header to give clients the ability to fetch pre-calculated results for a specified period (up to 12 hours), or explicitly request an ad hoc calculation.
    a. By default, all requests will be stored for 12 hours.
2. Updates to the calculation parameters (POST and PUT body) to give clients the option to choose between the different response formats offered with STACH v2 (for more information on STACH v2, refer to the documentation here and here).
3. Updated endpoint URLs to be engine-specific.
    a. v3: POST `analytics/engines/{engine[1]}/v3/calculations`
    b. v2: POST `analytics/engines/v2/calculations`
4. Support for user-provided calculation ids for polling and results URL construction via a new PUT method.
    a. E.g. You can now create a calculation request and specify a custom "daily_report" calculation id via the new PUT endpoint. This will allow you to directly construct the corresponding status polling and result pickup URLs:
        i. https://api.factset.com/analytics/engines/{engine}/v3/calculations/daily_report/status
        ii. https://api.factset.com/analytics/engines/{engine}/v3/calculations/daily_report/units/{unitId}/result

[1] Where engine can be pa, spar, pub, vault, fi, fiab, axp, bpm, afi, or npo.

5. Support for interactive endpoints that return calculation results as a response to the initial POST/PUT request if the calculation has 1 unit and it completes within a short period of time, defined per request.
6. Updated Java, Python, and .NET SDKs (coming soon) to support v3 API functionality.

3. Cache Control

v3 includes support for the cache-control max-stale header in the POST and PUT calculation endpoints.

3.1 What is Cache Control?

The Cache-Control HTTP header holds directives (instructions) for caching in both requests and responses. Engines APIs expose access to the max-stale directive to allow for calculation request caching. For more information on cache control, refer to the documentation here.

3.2 Using the Cache Control Header

Setting "max-stale=<staleness limit in seconds>" allows clients to fetch pre-calculated results with any subsequent POST requests[3] if they were last calculated within the staleness limit. The max cache-control value is "max-stale=43200" (12 hours).

Once set, the API will check to see if the stored results are within the staleness limit.

- If they are within the limit AND the request contains only 1 calculation unit:
  a. Then, the request will return a "201" response code with the results in the body of the POST/PUT response AND the status polling URL in the Location header. This allows for a quick response and eliminates unnecessary points accrual.
- If they are within the limit AND the request contains multiple calculation units:
  a. Then, the request will return a "200" response code with the result URLs in the body of the POST/PUT response. This allows for a quick response and eliminates unnecessary points accrual
- If the results are not within the staleness limit, a brand-new calculation request will be triggered to get the latest results. Note that the caching period starts after the individual calculation unit has completed

To immediately request the latest results, override the cache by setting "max-stale=0" in the Cache-control header parameter. ***Changes made to the underlying PA or SPAR document via the workstation will not trigger result re-calculation. The cache-control parameter should be set to "max-stale=0 to immediately request the latest results***

Note that the max-stale value only controls when a new calculation will be triggered with any subsequent POST or PUT requests (provided that the request parameters stay the same). All results generated will always be stored for 12 hours, the max-stale value will not affect this.

[3] The subsequent POST requests must have the same request body. Any changes in the request body will trigger a new calculation.

3.3 Default API Behavior

By default, all results are cached for 12 hours. This means that without sending any cache-control headers, after the first request successfully completes, all subsequent requests with unchanged request parameters will return the same results for 12 hours.

3.4 Setting Cache Control Via the Python SDK

- Create a String cache_control parameter and set to “max-stale=0”– line 31 below
- Pass cache_control parameter in the post_and_calculate() or put_and_calculate() methods – line 80 below
    - Note that all engines calculations APIs (PACalculationsApi, SPARCalculationsApi, VaultCalculationsApi, PubCalculationsApi, FICalculationsApi, FiabCalculationsApi) support the cache_control parameter via the post_and_calculate() and put_and_calculate() endpoints

```
28      # Create an instance of the API class
29      api_instance = pa_calculations_api.PACalculationsApi(api_client)
30      x_fact_set_api_long_running_deadline = 1 # int | Long running deadline in seconds when only one unit is passed in the PO
        body. (optional)
31      cache_control = "Cache-Control_example" # str | Standard HTTP header.  Accepts max-stale. (optional)
32      pa_calculation_parameters_root = PACalculationParametersRoot(
33  >       data={ ···
69          },
70          meta=CalculationMeta(
71              contentorganization="SimplifiedRow",
72              contenttype="Json",
73          ),
74      ) # PACalculationParametersRoot | Calculation Parameters (optional)
75
76      # example passing only required values which don't have defaults set
77      # and optional values
78      try:
79          # Create and Run PA calculation
80          api_response = api_instance.post_and_calculate
            (x_fact_set_api_long_running_deadline=x_fact_set_api_long_running_deadline, cache_control=cache_control,
            pa_calculation_parameters_root=pa_calculation_parameters_root)
81          pprint(api_response)
82      except fds.analyticsapi.engines.ApiException as e:
83          print("Exception when calling PACalculationsApi->post_and_calculate: %s\n" % e)
```

https://github.com/factset/analyticsapi-engines-python-sdk/blob/master/auto-generated-sdk/docs/PACalculationsApi.md#postandcalculate

3.5 Standalone Request Example

Standalone Request:

```
POST https://api.factset.com/analytics/engines/pa/v3/calculations

Request Headers:
Content-Type: application/json
Cache-Control: max-stale=0
Authorization: Basic ****************************
Accept: */*
Host: api.factset.com
Accept-Encoding: gzip, deflate, br
Content-Length: 739

Body:
{

  "data": {
    "1": {
      "componentid": "125872E5D836683A3EB09EF8C73D3E1BE854BE4EF6173BAB4BFBB5752788F4E9",
      "accounts": [
        {
          "id": "BENCH:SP50",
          "holdingsmode": "B&H",
        }
      ],
      "dates": {
        "startdate": "20170101",
        "enddate": "20180101",
        "frequency": "Monthly"
      },
      "groups": [
        {
          "id": "E5DB59F608778844CD784DD5659F65AA8A0A9F63F9E4314B2D92761AF0B1B3D9"
        }
      ],
      "benchmarks": [
        {
          "id": "BENCH:R.1000"
        }
      ],
      "currencyisocode": "USD"
    }
}
```

3.6 Sample Use Cases

Portfolio Manager looks at daily report on internal portal at 1pm

- POST calculation parameters for a multi-unit calculation to `analytics/engines/spar/v3/calculations` and set “max-stale=43200” (12-hour request cache)
    - A new calculation request is initiated, and the calculation successfully completes
    - User receives a 202, polls the status endpoint, and fetches results from the results endpoint
- That afternoon, user sends another POST to /analytics/engines/spar/v3/calculations with the same request body
    - A new calculation request is not initiated because the report was run within the past 12 hours
    - User receives a 200 and fetches results from the result URLs returned in the POST response body
- The next day, user sends another POST to `analytics/engines/spar/v3/calculations` with the same request body
    - A new calculation request is initiated because the report was last run more than 12 hours ago
    - User receives a 202, polls the status endpoint, and fetches results from the results endpoint

3 hours later, a new column is added to the report via the SPAR UI

- POST calculation parameters to `analytics/engines/spar/v3/calculations` calculations and set “max-stale=0” to receive the latest results
    - A new calculation request is initiated, and the calculation successfully completes
    - User receives a 202, polls the status endpoint, and fetches results from the results endpoint
    - Results include the newly-created column

4. Changes to Calculation Parameters Body

4.1 “data” Top-Level Object

The existing POST body and the new PUT request body will move under a new top-level object, “data”, replacing the engine-specific object.
In the example below, you can see the “pa” object in line 2 has been replaced with “data”.

4.2 “meta” Top-Level Object (Optional)

A new optional “meta” object will be available for clients to specify the response’s STACH v2 format. The “meta” object includes a “stachContentorganization” parameter to specify the STACH v2 format to return in the response, and a “format” parameter that allows user to specify whether the response should be in Json or Binary.

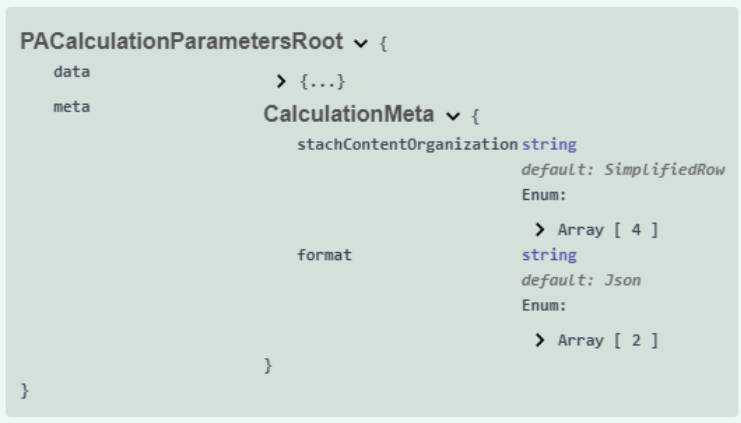
4.3 STACH v2

v2 of STACH introduced support for the new row organized format and simplified row format, along with the column organized format supported in the previous version of STACH. For more information on the new STACH v2 format in general, refer to the official documentation here. For Engines-specific documentation and samples, refer to the documentation here.

Note that STACH v2 will be the only version of STACH supported by the v3 Engines APIs, however STACH v2 supports the previous STACH v1 column organized format via the “stachContentorganization” parameter in the “meta” object.

Note that if the “meta” object is not specified, the default stachContentOrganization is SimplifiedRow and the default format is Json (STACH).

There is no other change in the calculation parameters from v2 to v3. The screenshot below is an excerpt from the PA Engine API Swagger hosted on Developer Portal. It shows the PA calculation parameter schema that describes the PA request body required and optional fields.



4.4 SDK Changes

4.4.1 CalculationParametersRoot object

In previous versions of the SDK, calculation parameters were constructed by creating an <Engine>CalculationParameters object.

In v5.0.0 of the Engines SDK, the <Engine>CalculationParameters object is replaced by the <Engine>CalculationParametersRoot object (line 32 in the screenshot below) which contains the new “data” and “meta” objects and all the relevant models.

```python
# Enter a context with an instance of the API client
with fds.analyticsapi.engines.ApiClient(configuration) as api_client:
    # Create an instance of the API class
    api_instance = pa_calculations_api.PACalculationsApi(api_client)
    x_fact_set_api_long_running_deadline = 1 # int | Long running deadline in seconds when only one unit is passed in the POST body. (optional)
    cache_control = "Cache-Control_example" # str | Standard HTTP header.  Accepts max-stale. (optional)
    pa_calculation_parameters_root = PACalculationParametersRoot(
        data={…
        },
        meta=CalculationMeta(
            contentorganization="SimplifiedRow",
            contenttype="Json",
        ),
    ) # PACalculationParametersRoot | Calculation Parameters (optional)

    # example passing only required values which don't have defaults set
    # and optional values
    try:
        # Create and Run PA calculation
        api_response = api_instance.post_and_calculate(x_fact_set_api_long_running_deadline=x_fact_set_api_long_running_deadline, cache_control=cache_control, pa_calculation_parameters_root=pa_calculation_parameters_root)
        pprint(api_response)
    except fds.analyticsapi.engines.ApiException as e:
        print("Exception when calling PACalculationsApi->post_and_calculate: %s\n" % e)
```

4.4.2 STACH Extensions SDK

The STACH Extensions SDK has been updated to include support for converting STACH v2 to Table format. The Analytics API Engines Python SDK v5.0.0 includes the latest STACH Extensions SDK, so only one dependency is required.

The code snippet below shows how to use the STACH extensions library to convert both STACH v2 and v1 to Table format.

```
# Stach v2 Row Organized format
stachBuilder = StachExtensionFactory.get_row_organized_builder(StachVersion.V2)
stachExtension = stachBuilder.set_package(data).build()  # data is the stach input in string or object format
dataFramesList = stachExtension.convert_to_dataframe()


# Stach v2 Column Organized format
stachBuilder = StachExtensionFactory.get_column_organized_builder(StachVersion.V2)
stachExtension = stachBuilder.set_package(data).build()  # data is the stach input in string or object format
dataFramesList = stachExtension.convert_to_dataframe()


# Stach v1 Column Organized format
stachBuilder = StachExtensionFactory.get_column_organized_builder(StachVersion.V1)
stachExtension = stachBuilder.set_package(data).build()  # data is the stach input in string or object format
dataFramesList = stachExtension.convert_to_dataframe()
```

https://github.com/factset/stach-extensions#usage

5. New and Modified Endpoints

5.1 New Endpoints

5.1.1 PUT and Calculate

PUT `analytics/engines/{engine}/v3/calculations/{id}`

- New method that allows clients to create and update custom calculation ids. It creates/updates and runs the engine calculation specified in the request body.
- `Cache-control` header functionality as outlined above.
- “data” and “meta” top level objects as outlined above.
- Interactive endpoint functionality as outlined here
- SDK method and models:
    - New put_and_calculate() method that takes in a String id (required), an <Engine>CalculationParametersRoot object (required), a cache_control String (optional), and a xFactSetApiLongRunningDeadline Integer (optional)
    - Response Object models:
        - 202 Response: CalculationStatusRoot
        - 200 Response: CalculationStatusRoot
        - 201 Response: ObjectRoot
    - You can find engine-specific documentation in the links below:

API-Specific Method	Request Object Model
PACalculationsApi.put_and_calculate()	PACalculationParametersRoot

SPARCalculationsApi.put_and_calculate()	SPARCalculationParametersRoot
VaultCalculationsApi.put_and_calculate()	VaultCalculationParametersRoot
PubCalculationsApi.put_and_calculate()	PubCalculationParametersRoot
FICalculationsApi.put_and_calculate()	FICalculationParametersRoot
FPOOptimizerApi.put_and_optimize()	FPOOptimizationParametersRoot
AXPOptimizeApi.put_and_optimize()	BPMOptimizationParametersRoot
BPMOptimizerApi.put_and_optimize()	AxiomaEquityOptimizationParametersRoot

Sample Use cases:

Portfolio Manager looks at a daily report on an internal portal at 9am

- PUT calculation parameters for a multi-unit calculation to `analytics/engines/pa/v3/calculations/daily_report` and set “max-stale=43200” (12-hour request cache)
  - A new calculation request is initiated, and the calculation successfully completes. The calculation parameters are successfully saved to the ‘daily_report’ calculation id.
  - User receives a 202
  - User polls the pre-constructed status URL:
    - https://api.factset.com/analytics/engines/pa/v3/calculations/daily_report/status
  - User fetches results from the pre-constructed results URL:
    - https://api.factset.com/analytics/engines/pa/v3/calculations/daily_report/units/{unitId}/result

Two hours later, the report is updated to include a new benchmark

- PUT calculation parameters to `analytics/engines/pa/v3/calculations/daily_report`
- and set “max-stale=43200” (12-hour request cache)
  - *A new calculation request is initiated since the request parameters have changed*, and the calculation successfully completes. The calculation parameters are successfully updated to the ‘daily_report’ calculation id.
  - User receives a 202, polls the pre-constructed status URL, and fetches results from the pre-constructed results URL.
  - Results include the newly-added benchmark data.
- That afternoon, user sends another PUT to `analytics/engines/pa/v3/calculations/daily_report` with the same request body
  - A new calculation request is not initiated because the report was run within the past 12 hours and the request parameters are unchanged.
  - User receives a 200 and fetches stored results from the pre-constructed result URLs returned in the POST response body.

- The next day, user sends another PUT to `analytics/engines/pa/v3/calculations/daily_report` with the same request body
    - A new calculation request is initiated because the report was last run more than 12 hours ago.
    - User receives a 202, polls the pre-constructed status URL, and fetches results from the pre-constructed results URL.

5.1.2 GET Status By ID

GET `analytics/engines/{engine}/v3/calculations/{id}/status`

- New endpoint to get calculation status, will function the same as GET {engine}/calculations/{id} did in v2
- If the calculation has finished computing, the location header will point to the result URL. Otherwise, if the calculation is still running and the X-FactSet-Api-PickUp-Progress header will contain a progress percentage.
- Response will include a parent object “data”
- SDK method and models:
    - New get_calculation_status_by_id() method that takes in a String id (required)
    - CalculationStatusRoot response object model for all response codes
    - You can find engine-specific documentation in the links below:

API-Specific Method
PACalculationsApi.get_calculation_status_by_id()
SPARCalculationsApi.get_calculation_status_by_id()
VaultCalculationsApi.get_calculation_status_by_id()
PubCalculationsApi.get_calculation_status_by_id()
FICalculationsApi.get_calculation_status_by_id()
FPOOptimizerApi.get_optimization_status_by_id()
AXPOptimizeApi.get_optimization_status_by_id()
BPMOptimizerApi.get_optimization_status_by_id()

5.2 Modified Endpoints

5.2.1 POST Calculation

POST `analytics/engines/{engine}/v3/calculations`

- `Cache-control` header functionality as outlined above.
- “data” and “meta” top level objects as outlined above.
- New response object models
- Interactive endpoint functionality as outlined here

- SDK changes:
    - New post_and_calculate() method that takes in an <Engine>CalculationParametersRoot object (required), a cache_control String (optional), and a xFactSetApiLongRunningDeadline Integer (optional)
        - Replaces the run_pa_calculation() method in the previous major version of the SDK
    - Response Object models:
        - 202 Response: CalculationStatusRoot
        - 200 Response: CalculationStatusRoot
        - 201 Response: ObjectRoot
    - You can find engine-specific documentation in the links below:

API-Specific Method	Request Object Model
PACalculationsApi.post_and_calculate()	PACalculationParametersRoot
SPARCalculationsApi.post_and_calculate()	SPARCalculationParametersRoot
VaultCalculationsApi.post_and_calculate()	VaultCalculationParametersRoot
PubCalculationsApi.post_and_calculate()	PubCalculationParametersRoot
FICalculationsApi.post_and_calculate()	FICalculationParametersRoot
FPOOptimizerApi.post_and_optimize()	FPOOptimizationParametersRoot
AXPOptimizeApi.post_and_optimize()	BPMOptimizationParametersRoot
BPMOptimizerApi.post_and_optimize()	AxiomaEquityOptimizationParametersRoot

5.2.2 GET Calculation Parameters By ID

GET `analytics/engines/{engine}/v3/calculations/{id}`

- Now returns calculation parameters instead of calculation status
- Takes in a calculation id string as a path parameter
- The calculation id is user-defined when created via the PUT request
- Calculation id is always returned in:
    - The location header of the POST/PUT response
    - The body of a successful POST/PUT request
- SDK changes:
    - New get_calculation_parameters() method that takes in a String id (required)
    - New Response Object models: engine-specific CalculationParametersRoot object models

- You can find engine-specific documentation in the links below:

API-Specific Method	New Response Object Model
PACalculationsApi.get_calculation_parameters()	PACalculationParametersRoot
SparCalculationsApi.get_calculation_parameters()	SPARCalculationParametersRoot
VaultCalculationsApi.get_calculation_parameters()	VaultCalculationParametersRoot
PubCalculationsApi.get_calculation_parameters()	PubCalculationParametersRoot
FICalculationsApi.get_calculation_parameters()	FiCalculationParametersRoot
FPOOptimizerApi.get_optimization_parameters()	FPOOptimizationParametersRoot
AXPOptimizerApi.get_optimization_parameters()	AxiomaEquityOptimizationParametersRoot
BPMOptimizerApi.get_optimization_parameters()	BPMOptimizationParametersRoot

5.2.3 GET Calculation Unit Results By ID

GET `analytics/engines/{engine}/v3/calculations/{id}/units/{unitId}/result`

- Functionally the same as v2, with result longevity dictated by PUT/POST cache-control headers
- Added new SDK method to support result retrieval
- Response will include a parent object “data”
- New URL format:
  - V2: /analytics/engines/**v2**/calculations/{id}/units/{unitid}/result
  - V3: /analytics/engines/**pa**/**v3**/calculations/{id}/units/{unitid}/result
- SDK changes:
  - New get_calculation_unit)_result_by_id() method that takes in a String id (required), and a String unitId (required)
  - Response Object model: ObjectRoot
  - You can find engine-specific documentation in the links below:

API-Specific Method
PACalculationsApi.get_calculation_unit_result_by_id()
SparCalculationsApi.get_calculation_unit_result_by_id()
VaultCalculationsApi.get_calculation_unit_result_by_id()
PubCalculationsApi.get_calculation_unit_result_by_id()
FICalculationsApi.get_calculation_result()
FPOOptimizerApi.get_optimization_result()
BpmOptimizerApi.get_optimization_result()

5.2.4 Components Lookup

GET `analytics/engines/{engine}/v3/components`

- V2: analytics/lookups/v2/engines/pa/components
- V3: analytics/engines/pa/v3/components
- SDK changes:
  - No change in the method names
  - Changes in the response object models:

Engine-Specific Method	New response object model
ComponentsApi.get_pa_component_by_id()	PAComponentRoot
ComponentsApi.get_pa_components()	ComponentSummaryRoot
ComponentsApi.get_spar_components()	ComponentSummaryRoot
ComponentsApi.get_vault_components()	ComponentSummaryRoot
ComponentsApi.get_vault_component_by_id()	VaultComponentRoot

5.2.5 Documents Lookup

GET `analytics/engines/{engine}/v3/document/{path}`

- V2: analytics/lookups/v2/engines/{engine}/documents/{path}
- V3: analytics/engines/{engine}/v3/documents/{path}
- SDK changes:
  - No change in the method name
  - New Response Object model: AccountDirectoriesRoot

5.2.6 Frequency Lookup

GET `analytics/engines/{engine}/v3/frequencies`

- V2: analytics/lookups/v2/engines/{engine}/frequencies
- V3: analytics/engines/{engine}/v3/frequencies
- SDK changes:
  - No change in the method name
  - New Response Object model: FrequencyRoot

5.2.7 Dates Lookup

GET `analytics/engines/{engine}/v3/dates`

- V2: analytics/lookups/v2/engines/{engine}/dates
- V3: analytics/engines/{engine}/v3/dates
- SDK changes:
  - No change in the method name
  - New Response Object model: DateParametersSummaryRoot

5.2.8 PA Columns Lookup

GET `analytics/engines/pa/v3/columns/{id}`

- V2: analytics/lookups/v2/engines/pa/columns/
- V3: analytics/engines/pa/v3/columns
- SDK changes:
  - No change in the method name
  - New Response Object model:
    - get_pa_column_by_id(): ColumnRoot
    - get_pa_columns(): ColumnSummaryRoot

5.2.9 PA Groups Lookup

GET `analytics/engines/pa/v3/groups`

- V2: analytics/lookups/v2/engines/pa/groups
- V3: analytics/engines/pa/v3/groups
- SDK changes:
  - No change in the method name
  - New Response Object model: GroupRoot

5.2.10 PA Column Statistics Lookup

GET `analytics/engines/pa/v3/columnstatistics`

- V2: analytics/lookups/v2/engines/pa/columnstatistics
- V3: analytics/engines/pa/v3/columnstatistics
- SDK changes:
  - No change in the method name
  - New Response Object model: ColumnStatisticRoot

5.2.11 Accounts Lookup

GET `analytics/lookups/v3/accounts`

- V2: analytics/lookups/v2/accounts/
- V3: analytics/lookups/v3/accounts/
- SDK changes:
  - No change in the method name
  - New Response Object model: AccountDirectoriesRoot

5.2.12 Currencies Lookup

GET `analytics/lookups/v3/currencies`

- V2: analytics/lookups/v2/engines/pa/currencies
- V3: analytics/lookups/v3/currencies

- New SDK method and models:
  - New get_currencies() method
    - Replaces the previous get_pa_currencies() method in the previous major version of the SDK
  - New response Object model: CurrencyRoot

6. Interactive Endpoints

V3 Includes support for interactive POST and PUT endpoints. These will return calculation results directly as a response to the initial POST/PUT request if the calculation has 1 unit and it completes within a specified period defined per request (20 seconds or less). Results will still be available via the

GET `analytics/engines/{engine}/v3/calculations/{id}/units/{unitId}/result` endpoint

- If the calculation completes successfully, a user receives a 201-response code and fetches results directly] in the POST response body.
- If the calculation completes with an error, a user receives a 200-response code and receives error information in the response body

6.1 Setting X-FactSet-Api-Long-Running-Deadline header Via the Python SDK

To set the X-FactSet-Api-Long-Running-Deadline header and specify the period that will result in an interactive endpoint response, you can do the following:

- Create an Integer xFactSetApiLongRunningDeadline parameter and set to desired amount of time – line 30 below
- Pass xFactSetApiLongRunningDeadline parameter in the post_and_calculate() or put_and_calculate() methods – line 80 below
  - Note that all engines calculations APIs (PaCalculationsApi, SparCalculationsApi, VaultCalculationsApi, PubCalculationsApi, FiCalculationsApi, FiabCalculationsApi) support the xFactSetApiLongRunningDeadline parameter via the postAndCalculate() and putAndCalculate() endpoints
- In this example, the X-FactSet-Api-Long-Running-Deadline has been set to 56 seconds, meaning that if the calculation takes less than 10 seconds to compute, it will return a 202 with the results directly (without needing to poll the status and fetch the results in separate calls)

```python
# Enter a context with an instance of the API client
with fds.analyticsapi.engines.ApiClient(configuration) as api_client:
    # Create an instance of the API class
    api_instance = pa_calculations_api.PACalculationsApi(api_client)
    x_fact_set_api_long_running_deadline = 1 # int | Long running deadline in seconds when only one unit is passed in the POST body. (optional)
    cache_control = "Cache-Control_example" # str | Standard HTTP header.  Accepts max-stale. (optional)
    pa_calculation_parameters_root = PACalculationParametersRoot(
        data={ ···
        },
        meta=CalculationMeta(
            contentorganization="SimplifiedRow",
            contenttype="Json",
        ),
    ) # PACalculationParametersRoot | Calculation Parameters (optional)

    # example passing only required values which don't have defaults set
    # and optional values
    try:
        # Create and Run PA calculation
        api_response = api_instance.post_and_calculate
        (x_fact_set_api_long_running_deadline=x_fact_set_api_long_running_deadline, cache_control=cache_control,
        pa_calculation_parameters_root=pa_calculation_parameters_root)
        pprint(api_response)
    except fds.analyticsapi.engines.ApiException as e:
        print("Exception when calling PACalculationsApi->post_and_calculate: %s\n" % e)
```

https://github.com/factset/analyticsapi-engines-python-sdk/blob/master/auto-generated-sdk/docs/PACalculationsApi.md#postandcalculate

6.2 Standalone Request Example

Standalone Request:

```
POST https://api.factset.com/analytics/engines/pa/v3/calculations
```

Request Headers:

```
Content-Type: application/json
Cache-Control: max-stale=0
X-FactSet-Api-Long-Running-Deadline: 10
Authorization: Basic ******************************
Accept: */*
Host: api.factset.com
Accept-Encoding: gzip, deflate, br
Content-Length: 739
```

Body:

```
{
```

```
  "data": {
    "1": {
      "componentid": "125872E5D836683A3EB09EF8C73D3E1BE854BE4EF6173BAB4BFBB5752788F4E9",
      "accounts": [
        {
          "id": "BENCH:SP50",
          "holdingsmode": "B&H",
        }
      ],
      "dates": {
        "startdate": "20170101",
        "enddate": "20180101",
        "frequency": "Monthly"
      },
      "groups": [
        {
          "id": "E5DB59F608778844CD784DD5659F65AA8A0A9F63F9E4314B2D92761AF0B1B3D9"
        }
      ],
      "benchmarks": [
        {
          "id": "BENCH:R.1000"
        }
      ],
      "currencyisocode": "USD"
    }
}
```


7. Installing v3-Compatible Python SDK

Follow the instructions outlined here to install the Analytics API Engines Python SDK v5.0.0 to your local Maven repository.