FACTSET ⟩ SEE THE ADVANTAGE

ENGINES API - VERSION 3.0

Java Migration Guide

November 2023

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 improves 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.
4. Support for OAuth2.0 which can be setup through FactSet's Developer Portal.

2. Summary of Changes

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

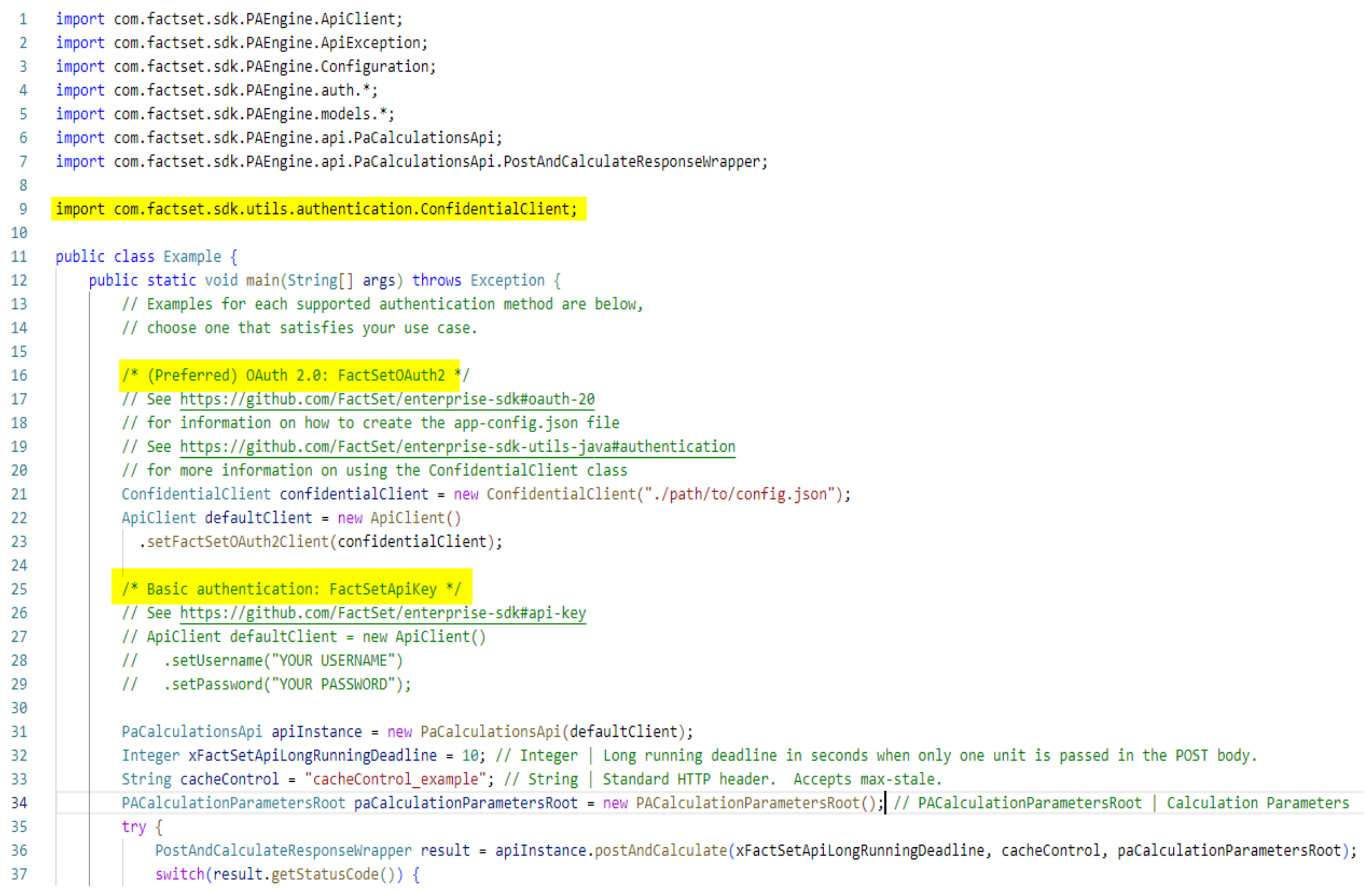
1. FactSet Enterprise SDK supports two forms of authentication -
    a. Basic Auth or API Key
    b. OAuth2.0
2. Support for cache-control header to give clients the ability to fetch pre-calculated results for a specified period (up to 12 hours), or explicitly request an ad hoc calculation.
    a. By default, all requests will be stored for 12 hours.
3. 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).
4. Updated endpoint URLs to be engine-specific.
    a. v3: POST `analytics/engines/`**{engine[1]}/v3**`/calculations`
    b. v2: POST `analytics/engines/v2/calculations`
5. Support for user-provided calculation ids for polling and result 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
6. 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.

[1] Where engine can be pa, spar, pub, vault, or fi.

3. Authentication

The FactSet Enterprise SDK supports two forms of authentication, both must be setup through FactSet's Developer Portal.

1. OAuth2.0 – (preferred).
2. API Key



```java
import com.factset.sdk.PAEngine.ApiClient;
import com.factset.sdk.PAEngine.ApiException;
import com.factset.sdk.PAEngine.Configuration;
import com.factset.sdk.PAEngine.auth.*;
import com.factset.sdk.PAEngine.models.*;
import com.factset.sdk.PAEngine.api.PaCalculationsApi;
import com.factset.sdk.PAEngine.api.PaCalculationsApi.PostAndCalculateResponseWrapper;

import com.factset.sdk.utils.authentication.ConfidentialClient;

public class Example {
    public static void main(String[] args) throws Exception {
        // Examples for each supported authentication method are below,
        // choose one that satisfies your use case.

        /* (Preferred) OAuth 2.0: FactSetOAuth2 */
        // See https://github.com/FactSet/enterprise-sdk#oauth-20
        // for information on how to create the app-config.json file
        // See https://github.com/FactSet/enterprise-sdk-utils-java#authentication
        // for more information on using the ConfidentialClient class
        ConfidentialClient confidentialClient = new ConfidentialClient("./path/to/config.json");
        ApiClient defaultClient = new ApiClient()
          .setFactSetOAuth2Client(confidentialClient);

        /* Basic authentication: FactSetApiKey */
        // See https://github.com/FactSet/enterprise-sdk#api-key
        // ApiClient defaultClient = new ApiClient()
        //   .setUsername("YOUR USERNAME")
        //   .setPassword("YOUR PASSWORD");

        PaCalculationsApi apiInstance = new PaCalculationsApi(defaultClient);
        Integer xFactSetApiLongRunningDeadline = 10; // Integer | Long running deadline in seconds when only one unit is passed in the POST body.
        String cacheControl = "cacheControl_example"; // String | Standard HTTP header.  Accepts max-stale.
        PACalculationParametersRoot paCalculationParametersRoot = new PACalculationParametersRoot(); // PACalculationParametersRoot | Calculation Parameters
        try {
            PostAndCalculateResponseWrapper result = apiInstance.postAndCalculate(xFactSetApiLongRunningDeadline, cacheControl, paCalculationParametersRoot);
            switch(result.getStatusCode()) {
```

4. Cache-Control

v3 includes support for the Cache-Control header in the POST and PUT calculation endpoints by accepting a max-stale value.

4.1 What is Cache-Control?

The Cache-Control HTTP header holds directives (instructions) for caching 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.

4.2 Using the Cache-Control Header

Setting “max-stale=<staleness limit in seconds>” allows clients to fetch pre-calculated results with any subsequent POST requests[2] 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.

4.3 Default API Behavior

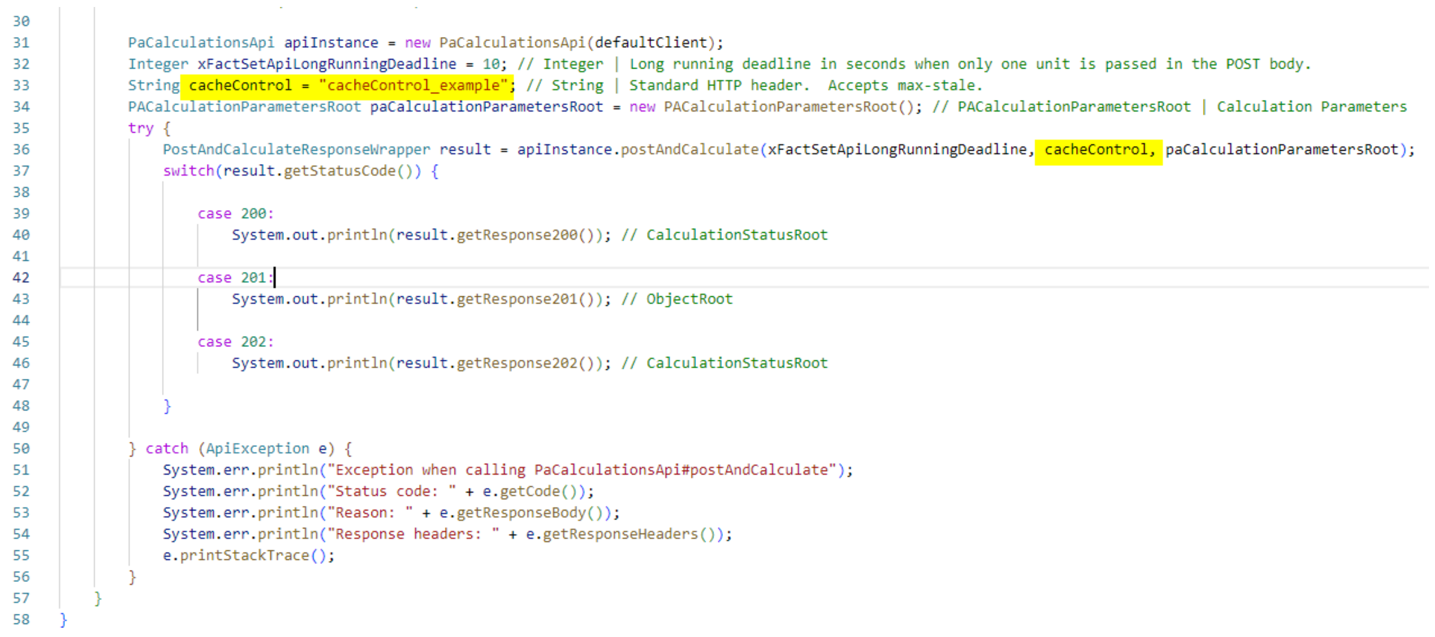
By default, all results are cached for 12 hours. This means that without sending any cache-control headers, after the first request successfully completes, all subsequent requests with unchanged request parameters will return the same results for 12 hours.

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

4.4 Setting Cache Control Via the Java SDK

- Create a String cacheControl parameter and set to “max-stale=0”– line 33 below
- Pass cacheControl parameter in the postAndCalculate() or putAndCalculate() methods – line 36 below
    - Note that all engines calculations APIs (PaCalculationsApi, SparCalculationsApi, VaultCalculationsApi, QuantCalculationsApi, FiCalculationsApi) support the cacheControl parameter via the postAndCalculate() and putAndCalculate() endpoints.

4.5 Standalone Request Example



```
30
31          PaCalculationsApi apiInstance = new PaCalculationsApi(defaultClient);
32          Integer xFactSetApiLongRunningDeadline = 10; // Integer | Long running deadline in seconds when only one unit is passed in the POST body.
33          String cacheControl = "cacheControl_example"; // String | Standard HTTP header.  Accepts max-stale.
34          PACalculationParametersRoot paCalculationParametersRoot = new PACalculationParametersRoot(); // PACalculationParametersRoot | Calculation Parameters
35          try {
36              PostAndCalculateResponseWrapper result = apiInstance.postAndCalculate(xFactSetApiLongRunningDeadline, cacheControl, paCalculationParametersRoot);
37              switch(result.getStatusCode()) {
38
39                  case 200:
40                      System.out.println(result.getResponse200()); // CalculationStatusRoot
41
42                  case 201:
43                      System.out.println(result.getResponse201()); // ObjectRoot
44
45                  case 202:
46                      System.out.println(result.getResponse202()); // CalculationStatusRoot
47
48              }
49
50          } catch (ApiException e) {
51              System.err.println("Exception when calling PaCalculationsApi#postAndCalculate");
52              System.err.println("Status code: " + e.getCode());
53              System.err.println("Reason: " + e.getResponseBody());
54              System.err.println("Response headers: " + e.getResponseHeaders());
55              e.printStackTrace();
56          }
57      }
58  }
```

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"
      }
  }
```

4.6 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 initially 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

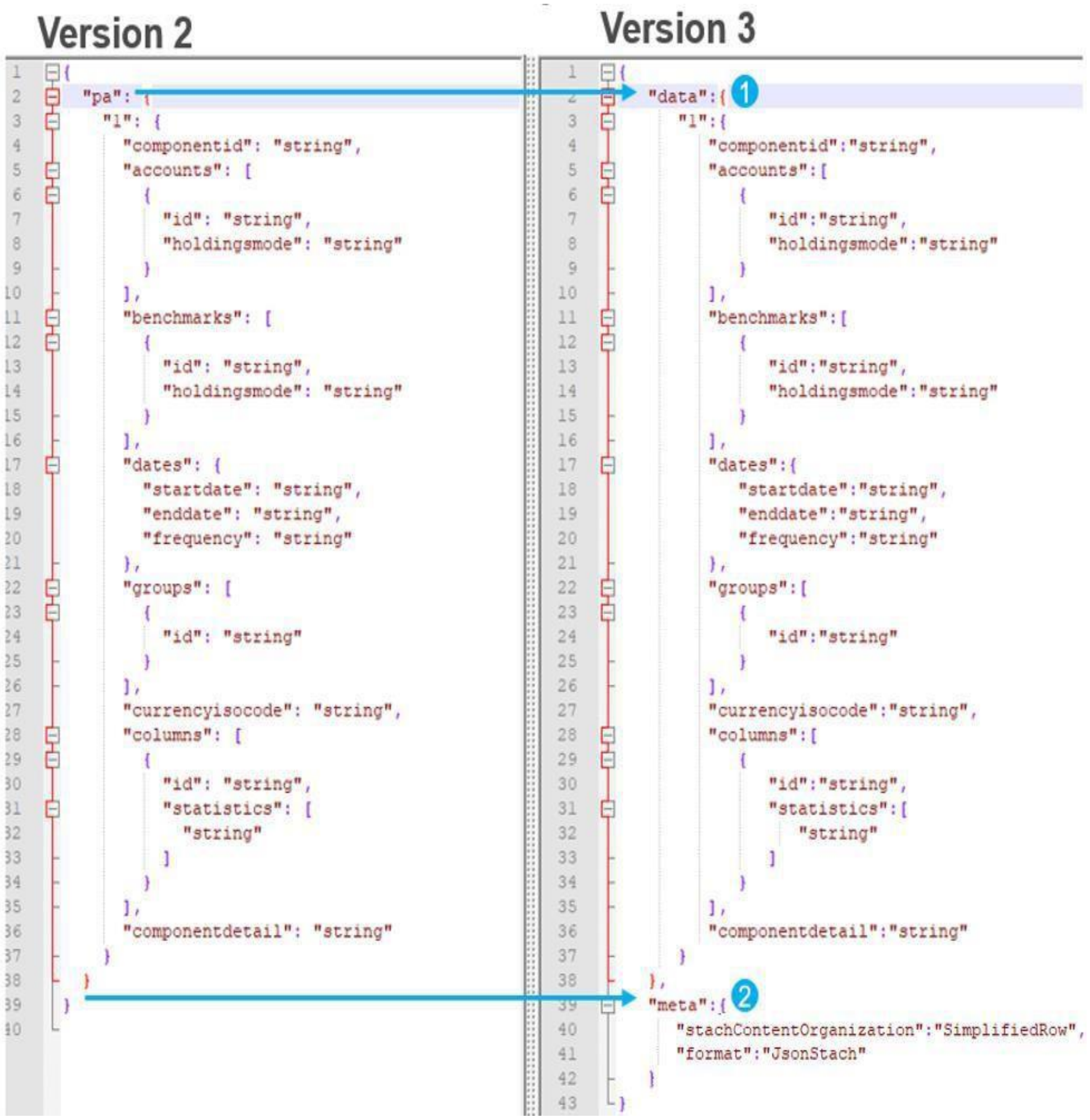
- POST calculation parameters to "analytics/engines/spar/v3/calculations" calculations and set "max-stale=0" to receive the latest results.
    - A new calculation request is initiated because the report has been updated.
    - User receives a 202, polls the status endpoint, and fetches results from the results endpoint.

5. Changes to Calculation Parameters Body

5.1 “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”.



Version 2

```
 1  {
 2    "pa": {
 3      "1": {
 4        "componentid": "string",
 5        "accounts": [
 6          {
 7            "id": "string",
 8            "holdingsmode": "string"
 9          }
10        ],
11        "benchmarks": [
12          {
13            "id": "string",
14            "holdingsmode": "string"
15          }
16        ],
17        "dates": {
18          "startdate": "string",
19          "enddate": "string",
20          "frequency": "string"
21        },
22        "groups": [
23          {
24            "id": "string"
25          }
26        ],
27        "currencyisocode": "string",
28        "columns": [
29          {
30            "id": "string",
31            "statistics": [
32              "string"
33            ]
34          }
35        ],
36        "componentdetail": "string"
37      }
38    }
39  }
40
```

Version 3

```
 1  {
 2    "data":{
 3      "1":{
 4        "componentid":"string",
 5        "accounts":[
 6          {
 7            "id":"string",
 8            "holdingsmode":"string"
 9          }
10        ],
11        "benchmarks":[
12          {
13            "id":"string",
14            "holdingsmode":"string"
15          }
16        ],
17        "dates":{
18          "startdate":"string",
19          "enddate":"string",
20          "frequency":"string"
21        },
22        "groups":[
23          {
24            "id":"string"
25          }
26        ],
27        "currencyisocode":"string",
28        "columns":[
29          {
30            "id":"string",
31            "statistics":[
32              "string"
33            ]
34          }
35        ],
36        "componentdetail":"string"
37      }
38    },
39    "meta":{
40      "stachContentOrganization":"SimplifiedRow",
41      "format":"JsonStach"
42    }
43  }
```

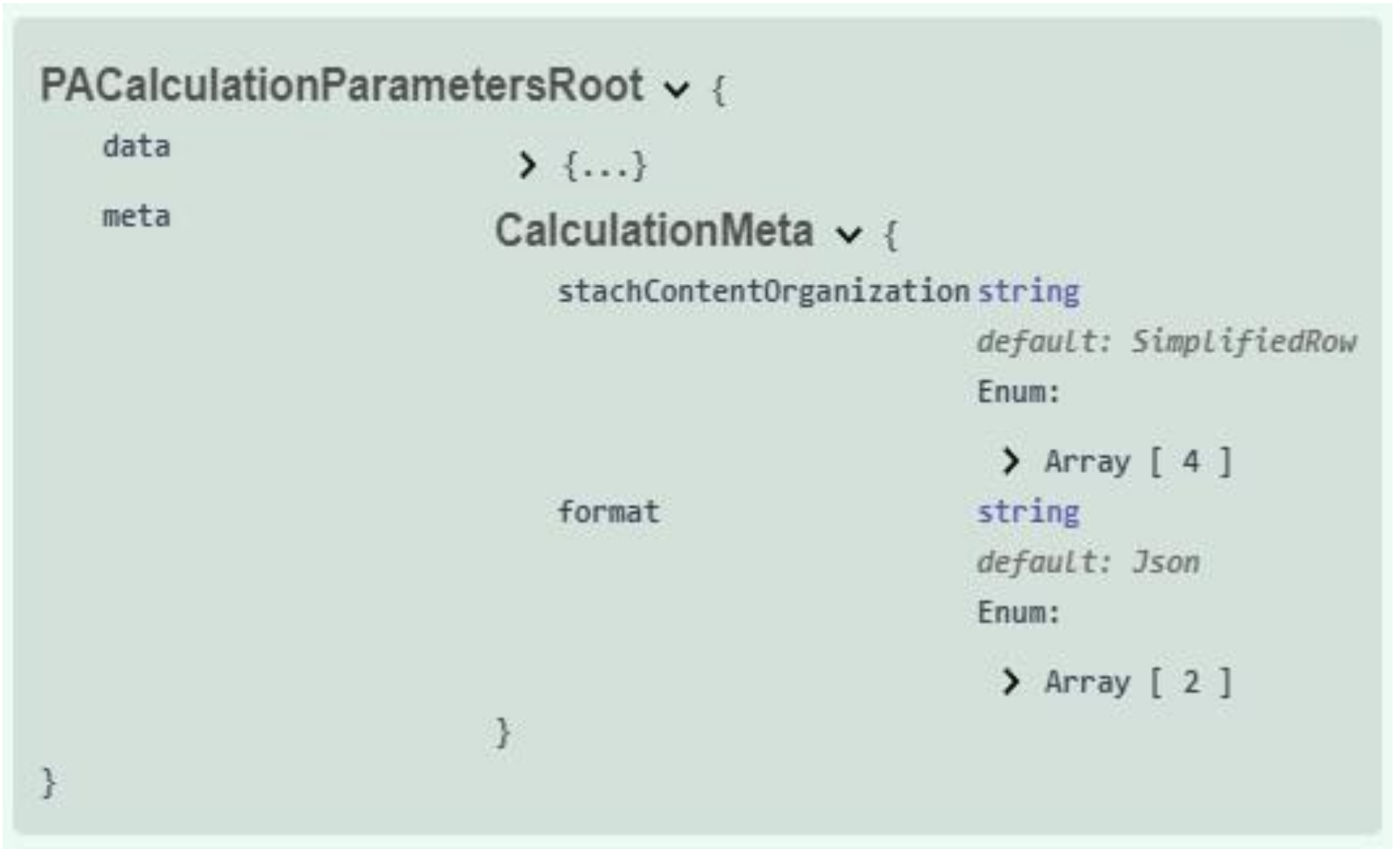
5.2 “meta” Top-Level Object (Optional)

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.

5.3 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



```
PACalculationParametersRoot ⌄ {
    data                        › {...}
    meta                        CalculationMeta ⌄ {
                                    stachContentOrganization string
                                                             default: SimplifiedRow
                                                             Enum:
                                                               › Array [ 4 ]
                                    format                   string
                                                             default: Json
                                                             Enum:
                                                               › Array [ 2 ]
                                }
}
```

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).

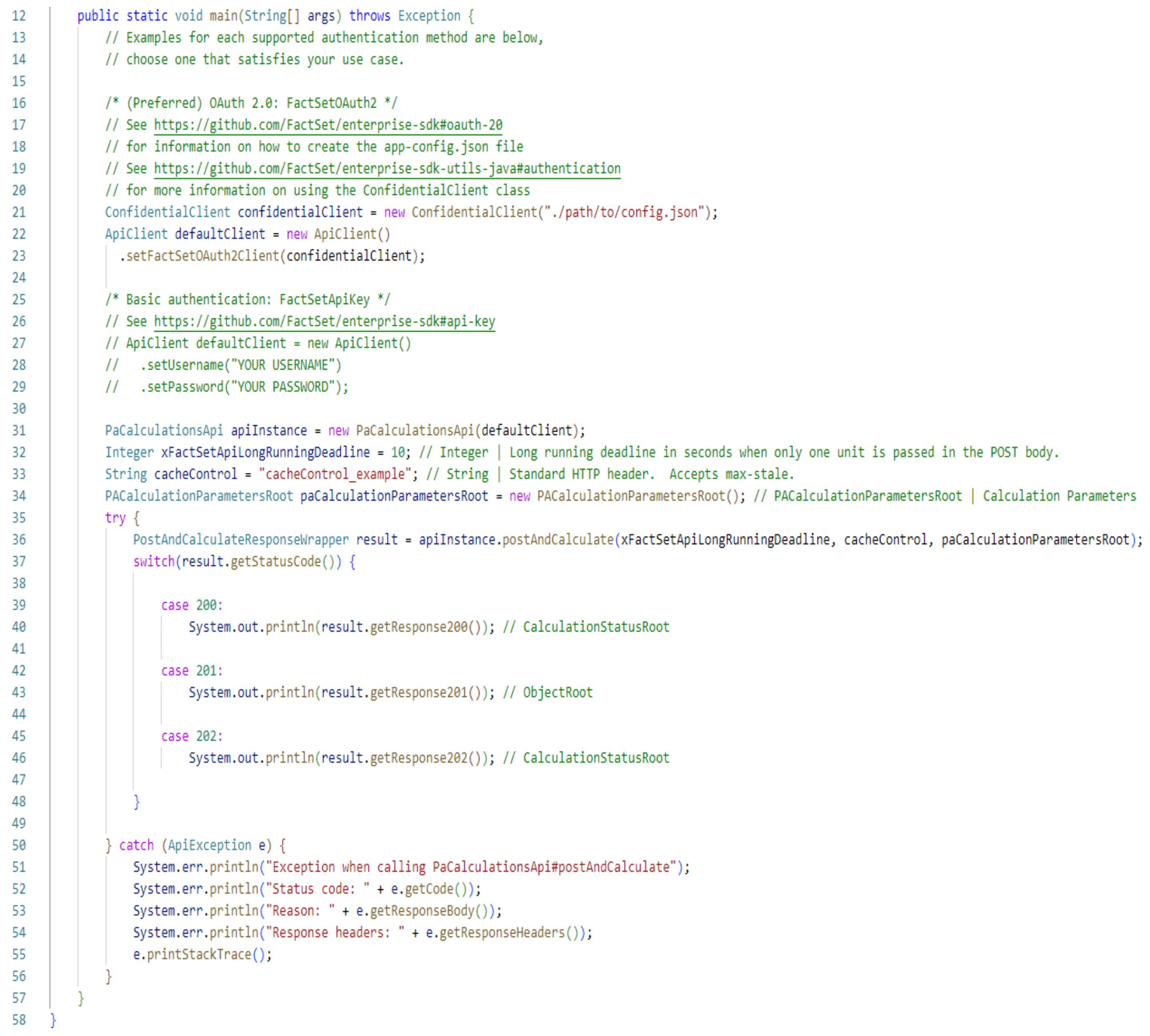
There is no other change in the calculation parameters from v2 to v3. The screenshot below is an excerpt from the PA Engine API Swagger hosted on Developer Portal. It shows the PA calculation parameter schema that describes the PA request body required and optional fields.

5.4 SDK Changes

5.4.1 CalculationParametersRoot object

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

In the latest version of the Engines SDK for v3, the <Engine>CalculationParameters object is replaced by the <Engine>**CalculationParametersRoot** object (line 34 in the screenshot below) which contains the new “**data**” and “**meta**” objects and all the relevant models.

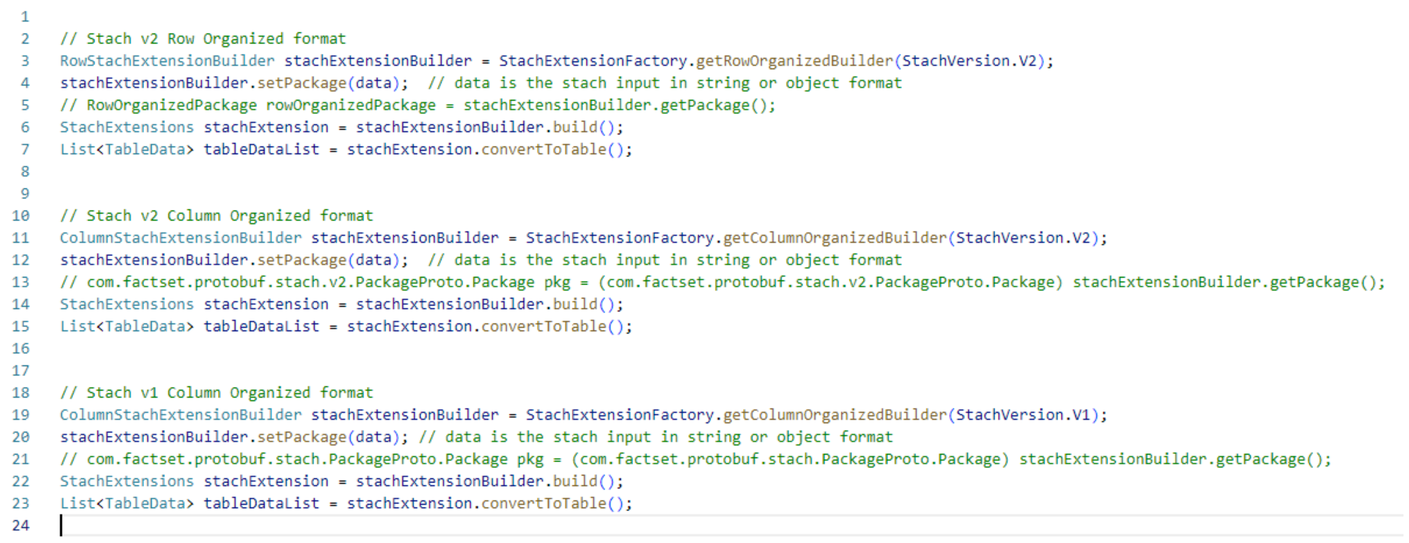


```
12      public static void main(String[] args) throws Exception {
13          // Examples for each supported authentication method are below,
14          // choose one that satisfies your use case.
15
16          /* (Preferred) OAuth 2.0: FactSetOAuth2 */
17          // See https://github.com/FactSet/enterprise-sdk#oauth-20
18          // for information on how to create the app-config.json file
19          // See https://github.com/FactSet/enterprise-sdk-utils-java#authentication
20          // for more information on using the ConfidentialClient class
21          ConfidentialClient confidentialClient = new ConfidentialClient("./path/to/config.json");
22          ApiClient defaultClient = new ApiClient()
23            .setFactSetOAuth2Client(confidentialClient);
24
25          /* Basic authentication: FactSetApiKey */
26          // See https://github.com/FactSet/enterprise-sdk#api-key
27          // ApiClient defaultClient = new ApiClient()
28          //   .setUsername("YOUR USERNAME")
29          //   .setPassword("YOUR PASSWORD");
30
31          PaCalculationsApi apiInstance = new PaCalculationsApi(defaultClient);
32          Integer xFactSetApiLongRunningDeadline = 10; // Integer | Long running deadline in seconds when only one unit is passed in the POST body.
33          String cacheControl = "cacheControl_example"; // String | Standard HTTP header.  Accepts max-stale.
34          PACalculationParametersRoot paCalculationParametersRoot = new PACalculationParametersRoot(); // PACalculationParametersRoot | Calculation Parameters
35          try {
36              PostAndCalculateResponseWrapper result = apiInstance.postAndCalculate(xFactSetApiLongRunningDeadline, cacheControl, paCalculationParametersRoot);
37              switch(result.getStatusCode()) {
38
39                  case 200:
40                      System.out.println(result.getResponse200()); // CalculationStatusRoot
41
42                  case 201:
43                      System.out.println(result.getResponse201()); // ObjectRoot
44
45                  case 202:
46                      System.out.println(result.getResponse202()); // CalculationStatusRoot
47
48              }
49
50          } catch (ApiException e) {
51              System.err.println("Exception when calling PaCalculationsApi#postAndCalculate");
52              System.err.println("Status code: " + e.getCode());
53              System.err.println("Reason: " + e.getResponseBody());
54              System.err.println("Response headers: " + e.getResponseHeaders());
55              e.printStackTrace();
56          }
57      }
58  }
```

5.4.2 STACH Extensions SDK

The STACH Extensions Package has been updated to include support for converting STACH v2 into a dataframe. The FactSet Enterprise Java SDK v1.0.0 for API Engines Java v1.0.0 does not include the latest STACH Extensions package, hence the user need to install the Java stach extensions package manually from here.

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
RowStachExtensionBuilder stachExtensionBuilder = StachExtensionFactory.getRowOrganizedBuilder(StachVersion.V2);
stachExtensionBuilder.setPackage(data);  // data is the stach input in string or object format
// RowOrganizedPackage rowOrganizedPackage = stachExtensionBuilder.getPackage();
StachExtensions stachExtension = stachExtensionBuilder.build();
List<TableData> tableDataList = stachExtension.convertToTable();


// Stach v2 Column Organized format
ColumnStachExtensionBuilder stachExtensionBuilder = StachExtensionFactory.getColumnOrganizedBuilder(StachVersion.V2);
stachExtensionBuilder.setPackage(data);  // data is the stach input in string or object format
// com.factset.protobuf.stach.v2.PackageProto.Package pkg = (com.factset.protobuf.stach.v2.PackageProto.Package) stachExtensionBuilder.getPackage();
StachExtensions stachExtension = stachExtensionBuilder.build();
List<TableData> tableDataList = stachExtension.convertToTable();


// Stach v1 Column Organized format
ColumnStachExtensionBuilder stachExtensionBuilder = StachExtensionFactory.getColumnOrganizedBuilder(StachVersion.V1);
stachExtensionBuilder.setPackage(data); // data is the stach input in string or object format
// com.factset.protobuf.stach.PackageProto.Package pkg = (com.factset.protobuf.stach.PackageProto.Package) stachExtensionBuilder.getPackage();
StachExtensions stachExtension = stachExtensionBuilder.build();
List<TableData> tableDataList = stachExtension.convertToTable();

```

https://github.com/factset/stach-extensions#java-1

6. New and Modified Endpoints

6.1 New Endpoints

6.1.1 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 putAndCalculate() method that takes in a String id (required), an <Engine>CalculationParametersRoot object (required), a cacheControl 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.putAndCalculate()	PaCalculationParametersRoot
SparCalculationsApi.putAndCalculate()	SparCalculationParametersRoot
VaultCalculationsApi.putAndCalculate()	VaultCalculationParametersRoot
QuantCalculationsApi.putAndCalculate()	QuantCalculationParametersRoot
FiCalculationsApi.putAndCalculate()	FICalculationParametersRoot

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.

6.1.2 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.
- Status field in the response points to the state of the calculation. It can take three values, “Queued”, “In Progress” and “Completed”.
- Response will include a parent object “data.”
- SDK method and models:
  - New getCalculationStatusById(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.getCalculationStatusById(id)
SparCalculationsApi.getCalculationStatusById(id)
VaultCalculationsApi.getCalculationStatusById(id)
QuantCalculationsApi.getCalculationStatusById(id)
FiCalculationsApi.getCalculationStatusById(id)

6.1.3 GET All Calculations

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

- New endpoint which returns all the calculations.
- V3: analytics/engines/pa/v3/calculations
- This endpoint is available only for PA, SPAR, Vault, and Quant Engine APIs.

API-Specific Method
PaCalculationsApi.getAllCalculations(pageNumber)
SparCalculationsApi.getAllCalculations(pageNumber)
VaultCalculationsApi.get.getAllCalculations(pageNumber)
QuantCalculationsApi.getAllCalculations(pageNumber)

6.1.4 PA Pricing Source Lookups

GET `/analytics/engines/pa/v3/pricing-sources`

- This endpoint lists all the PA pricing sources that can be applied to a PA calculation.
- V3: analytics/engines/pa/v3/pricing-sources
- API Specific Method - PricingSourcesApi.getPAPricingSources()
- New response Object model - PaPricingSourceRoot

6.1.5 GET Quant Engine Calculation Metadata Information by Id

GET `/analytics/engines/quant/v3/calculations/{id}/units/{unitId}/info`

- This endpoint is used to get the metadata information of a previously requested calculation.
- V3: analytics/engines/quant/v3/calculations/{id}/units/{unitId}/info

- API Specific Method - QuantCalculationsApi.getCalculationUnitInfoById

6.1.6 Fixed Income Calculations Discount Curve Lookups

GET `/analytics/engines/fi/v3/discount-curves`

- This endpoint lists all the discount curves that can be applied to a FI calculation.
- V3: analytics/engines/fi/v3/discount-curves
- API Specific Method - DiscountCurvesApi.getAllFIDiscountCurves
- New response Object model - FIDiscountCurveInfoRoot

6.2 Modified Endpoints

6.2.1 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 postAndCalculate() method that takes in an <Engine>CalculationParametersRoot object (required), a cacheControl String (optional), and a xFactSetApiLongRunningDeadline Integer (optional)
    - Replaces the runPACalculation() 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.postAndCalculate()	PaCalculationParametersRoot
SparCalculationsApi.postAndCalculate()	SparCalculationParametersRoot
VaultCalculationsApi.postAndCalculate()	VaultCalculationParametersRoot
QuantCalculationsApi.postAndCalculate()	QuantCalculationParametersRoot
FiCalculationsApi.postAndCalculate()	FICalculationParametersRoot

6.2.2 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 GetCalculationParameters() 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.getCalculationParameters()	PaCalculationParametersRoot
SparCalculationsApi.getCalculationParameters()	SparCalculationParametersRoot
VaultCalculationsApi.getCalculationParameters()	VaultCalculationParametersRoot
QuantCalculationsApi.getCalculationParameters()	QuantCalculationParametersRoot
FiCalculationsApi.getCalculationParameters()	FICalculationParametersRoot

6.2.3 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
  - For Fixed Income v3 - /analytics/engines**/fi/v3/**calculations/{id}/result
- SDK changes:
  - New getCalculationUnitResultById() 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.getCalculationUnitResultById()
SparCalculationsApi.getCalculationUnitResultById()
VaultCalculationsApi.getCalculationUnitResultById()
QuantCalculationsApi.getCalculationUnitResultById()
FiCalculationsApi.getCalculationResult()

6.2.4 DELETE Calculation By ID

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

- Functionally is same as v2, used to cancel a previously submitted calculation using the calculationId as input.

- A successful response will fetch a 204 status code.
- New URL format:
    - V2: /analytics/engines/**v2**/calculations/{id}
    - V3: /analytics/engines/**pa**/**v3**/calculations/{id}

API-Specific Method
PaCalculationsApi.cancelCalculationById(id)
SparCalculationsApi.cancelCalculationById(id)
VaultCalculationsApi.cancelCalculationById(id)
QuantCalculationsApi.cancelCalculationById(id)
FiCalculationsApi.cancelCalculationById(id)

6.2.5 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	API-Specific Method	New response object model
PA	ComponentsApi.getPAComponents()	ComponentSummaryRoot
	ComponentsApi.getPAComponentById()	PaComponentRoot
SPAR	ComponentsApi.getSPARComponents()	ComponentSummaryRoot
	ComponentsApi.getSPARComponentById()	SparComponentRoot
Vault	ComponentsApi.getVaultComponents()	ComponentSummaryRoot
	ComponentsApi.getVaultComponentById()	VaultComponentRoot

6.2.6 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: DocumentDirectoriesRoot

Engine	API-Specific Method	New response object model
PA	DocumentsApi.getPA3Documents()	DocumentDirectoriesRoot

SPAR	DocumentsApi.getSPAR3Documents()	DocumentDirectoriesRoot
Vault	DocumentsApi.getVaultDocuments()	DocumentDirectoriesRoot

6.2.7 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

Engine	API-Specific Method	New response object model
PA	FrequenciesApi.getPAFrequencies()	FrequencyRoot
SPAR	FrequenciesApi.getSPARFrequencies()	FrequencyRoot
Vault	FrequenciesApi.getVaultFrequencies()	FrequencyRoot

6.2.8 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

Engine	API-Specific Method	New response object model
PA	DatesApi.convertPADatesToAbsoluteFormat(enddate, componentid, account)	DateParametersSummaryRoot
Vault	DatesApi.convertVaultDatesToAbsoluteFormat(enddate, componentid, account)	DateParametersSummaryRoot

6.2.9 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:
    - getPAColumnById(): ColumnRoot
    - getPAColumns(): ColumnSummaryRoot

6.2.10 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:
    - getPAGroups(): GroupRoot
    - getPAGroupingFrequencies(): FrequencyRoot

6.2.11 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:
    - getPAColumnStatistics(): ColumnStatisticRoot

6.2.12 Vault Configurations Lookup

GET `/analytics/engines/vault/v3/configurations`

- V2: analytics/lookups/v2/engines/vault/configurations
- V3: analytics/engines/vault/v3/configurations
- SDK changes:
  - New Response Object model:
    - getVaultConfigurations(account): VaultConfigurationSummaryRoot
    - getVaultConfigurationById(id): VaultConfigurationRoot

6.2.13 SPAR Benchmarks Lookup

GET `/analytics/engines/spar/v3/benchmarks`

- V2: analytics/lookups/v2/engines/spar/benchmarks
- V3: analytics/engines/spar/v3/benchmarks
- SDK changes:
  - New Response Object model:
    - getSPARBenchmarkById(id): SPARBenchmarkRoot

6.2.14 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

Engine	API-Specific Method	New response object model
PA	AccountsApi.getAccounts(path)	AccountDirectoriesRoot
SPAR	AccountsApi.getAccounts(path)	AccountDirectoriesRoot
SPAR	AccountsApi.getSPARReturnsType(accountPath)	SPARAccountsRoot

6.2.15 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 getCurrencies() method
    - Replaces the previous getPACurrencies() method in the previous major version of the SDK
  - New response Object model: CurrencyRoot

Engine	API-Specific Method	New response object model
PA	CurrenciesApi.getCurrencies()	CurrencyRoot
SPAR	CurrenciesApi.getCurrencies()	CurrencyRoot

7. 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.

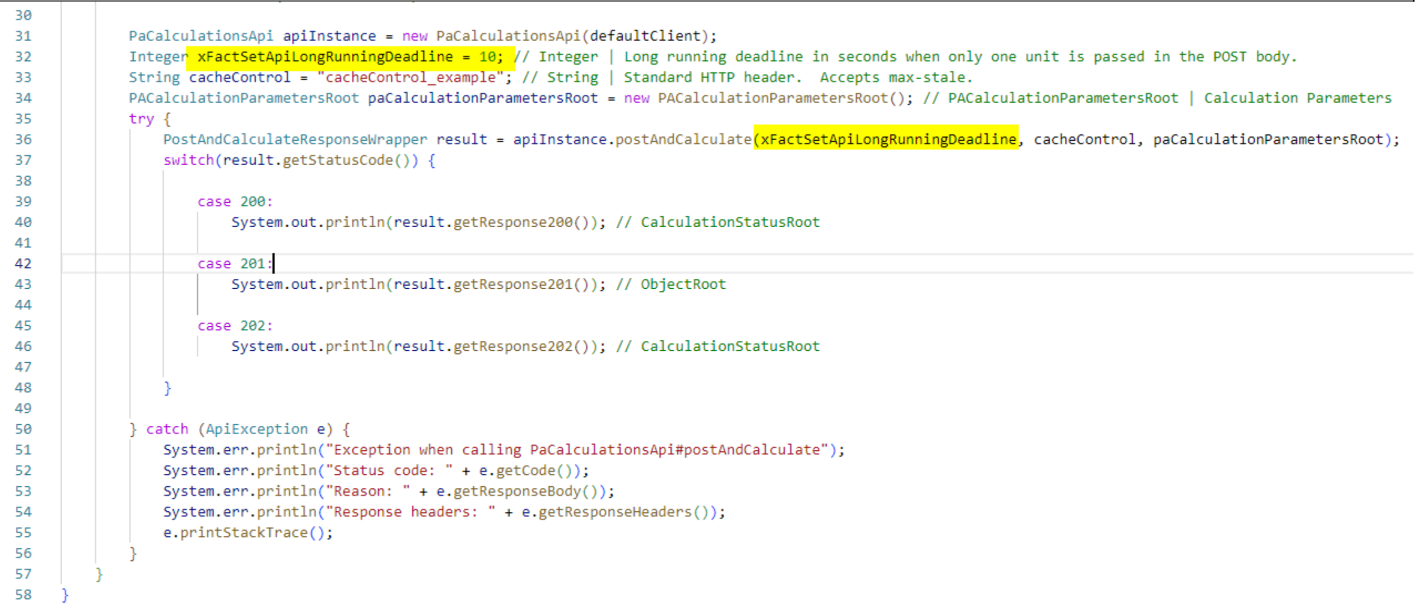
7.1 Setting X-FactSet-Api-Long-Running-Deadline header Via the Java SDK

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 32 below
- PAss xFactSetApiLongRunningDeadline parameter in the postAndCalculate() or putAndCalculate() methods – line 36 below
  - Note that all engines calculations APIs (PaCalculationsApi, SparCalculationsApi, VaultCalculationsApi, QuantCalculationsApi, FiCalculationsApi,) support the xFactSetApiLongRunningDeadline parameter via the postAndCalculate() and putAndCalculate() endpoints
- In this example, the X-FactSet-Api-Long-Running-Deadline has been set to 10 seconds, meaning that if

the calculation takes less than 10 seconds to compute, it will return a 201 with the results directly (without needing to poll the status and fetch the results in separate calls) and if the calculation is not completed within 10s, it will fetch a 202 and go into the long-running mode.

- User can then poll for the status of the calculation using the GET Status by Id endpoint and get the result using the GET Result by Id endpoint once the status is completed.



```java

        PaCalculationsApi apiInstance = new PaCalculationsApi(defaultClient);
        Integer xFactSetApiLongRunningDeadline = 10; // Integer | Long running deadline in seconds when only one unit is passed in the POST body.
        String cacheControl = "cacheControl_example"; // String | Standard HTTP header.  Accepts max-stale.
        PACalculationParametersRoot paCalculationParametersRoot = new PACalculationParametersRoot(); // PACalculationParametersRoot | Calculation Parameters
        try {
            PostAndCalculateResponseWrapper result = apiInstance.postAndCalculate(xFactSetApiLongRunningDeadline, cacheControl, paCalculationParametersRoot);
            switch(result.getStatusCode()) {

                case 200:
                    System.out.println(result.getResponse200()); // CalculationStatusRoot

                case 201:
                    System.out.println(result.getResponse201()); // ObjectRoot

                case 202:
                    System.out.println(result.getResponse202()); // CalculationStatusRoot

            }

        } catch (ApiException e) {
            System.err.println("Exception when calling PaCalculationsApi#postAndCalculate");
            System.err.println("Status code: " + e.getCode());
            System.err.println("Reason: " + e.getResponseBody());
            System.err.println("Response headers: " + e.getResponseHeaders());
            e.printStackTrace();
        }
    }
}
```

7.2 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"
    }
}
```

8. Installing FactSet Enterprise Java SDK

Follow the instructions outlined here to install the FactSet Enterprise Java SDKs.

Engine	SDK
PA	FactSet.SDK/PAEngine 1.0.0
SPAR	FactSet.SDK.SPAREngine 1.0.0
Vault	FactSet.SDK.Vault 1.0.0
Quant	FactSet.SDK.QuantEngine 1.0.0
FI	FactSet.SDK.FixedIncomeCalculation 1.0.0

For users currently on version-3 of the APIs and using **fds.analytics.sdk** can migrate to **fds.sdk.<Engine>.**

We encourage all existing users on **Analytics SDK v5.0.0** and above, to migrate over to **FactSet Enterprise SDK v1.0.0**.

All upcoming enhancements to the APIs will be supported only on **FactSet Enterprise SDKs.** Analytics SDKs will be put into sunset mode and will only be maintained to address any bugs identified.

9. Troubleshooting

The Please file an issue tracker with category **"Analytics APIs – Support"** and sub-category **"v3 Migration"** in case you need any assistance or have any feedback.

Please also note that we are targeting to retire the Analytics SDKs by the end of **August 2024.**

10. Appendix

10.1 CalculationStatusRoot

Engine	API-Specific Method
PA	CalculationStatusRoot
SPAR	CalculationStatusRoot
Vault	CalculationStatusRoot
Quant	CalculationStatusRoot

10.2 ObjectRoot

Engine	API-Specific Method
PA	ObjectRoot
SPAR	ObjectRoot
Vault	ObjectRoot
Quant	ObjectRoot
FI	ObjectRoot