Introduction

In Prowess, data can be extracted either by executing the &quot;Query Batches&quot; or by accessing &quot;Pre-

defined Reports.&quot;

The API facility allows users to automate the processes for executing Query batches and accessing

pre-defined reports.

The two API&#39;s that are available in Prowess are the Batch API and the Report API.

The output of the APIs are available in the following formats:

1. &#39;.txt&#39; file (Pipe delimited)

2. &#39;.json&#39; file

1. Batch API

Batch API is a deffered response API call. As executing a batch can become time consuming,

two different types of API calls are required to execute a Batch API.

What is a batch ?

A Batch is a command line or a set of commands that is generated when a user executes a query.

It is available in the Commands History and Planner in prowess.

To execute a Batch API, we suggest users to run queries for the data they want to extract and then

download the corresponding Batch file from the Commands History and Planner section.

The Batch file contains the commands in encrypted format. The Batch file needs to be downloaded

to make an API call.

1. To post the batch. (sendbatch)

In this case a batch posted is not executed immediately but queued. A unique token is sent in

response

which can be used by the users to check the status of the execution of a particular batch.

Users should note down (save) the token received as the same will be used in getting the

update for the batch executed.

2. To check the status of batch execution. (getbatch)

The token received in (1) is sent, the response is a zip file containing the data in (.txt) or (.json)

file(s).

If there is an error then a (.json) file containing the status is given in the response.

NOTE: The file in response is a (.zip) file, in case the batch is processed and ready for download.

Response Data:

 Response file (.zip) contains multiple (.txt) files and a single execution log

(TOKEN.lst).

 The (.txt) files are numbered e.g: 1.txt, 2.txt, etc.

 They correspond to all the output sheets / worksheets that were involved in the

execution of the batch.

 The (.json) format files are numbered e.g. 1.json, 2.json etc. The json file contains

three sections:

1. &quot;meta&quot; : which contains number of rows and columns and other metadata

2. &quot;head&quot; : which contains the headers and

3. &quot;data&quot; : which contains the output.

Note: (.json) output format is available only for &#39;OSC&#39; and &#39;WS&#39; outputs. If &#39;TXT&#39; output format is

selected then the output generated will be in a (.txt) format only.

1. SendBatch:

SendBatch Endpoint URL: https://prowess.cmie.com/api/sendbatch

SendBatch example using Shell-Curl:

curl -F apikey=YOUR\_API\_KEY \

-F batchfile=@YOUR\_BATCH\_FILE \

-X POST \

https://prowess.cmie.com/api/sendbatch -o -

NOTE: By default the output is pipe-delimited. For JSON add an additional post parameter

&quot;format=json&quot;.

SendBatch example using php:

&lt;?php

$ch= curl\_init();

curl\_setopt($ch, curlopt\_url, &#39;https://prowess.cmie.com/api/sendbatch&#39;);

curl\_setopt($ch, curlopt\_post, true);

curl\_setopt($ch, curlopt\_postfields, [ &#39;apikey&#39; =&gt; &#39;YOUR\_API\_KEY&#39;, &#39;batchfile&#39; =&gt; new

curlfile(&#39;YOUR\_BATCH\_FILE&#39;) ]);

$response= curl\_exec($ch);

if(curl\_errno($ch))

printf(&quot;error: %s\n&quot;, curl\_error($ch));

else

printf(&quot;response: %s\n&quot;, $response);

curl\_close($ch);

?&gt;

SendBatch example using Python:

import requests

url= &#39;https://prowess.cmie.com/api/sendbatch&#39;

mydata= { &#39;apikey&#39;: &#39;YOUR\_API\_KEY&#39;, &#39;format&#39;: &#39;json&#39; } # If you want the output in json format

myfile= { &#39;batchfile&#39;: open(&#39;YOUR\_BATCH\_FILE&#39;, &#39;rb&#39;) } # query batchfile

response= requests.post(url, data = mydata, files = myfile)

#print the response text (the content of the requested file):

print(response.text)

SendBatch response (JSON):

{

&quot;status&quot; : &quot;200&quot;,

&quot;token&quot; : &quot;20240227\_125569&quot;,

&quot;errcode&quot; : 0,

&quot;errdesc&quot; : &quot;Success&quot;,

&quot;servertime&quot; : &quot;27-02-24 16:48:32&quot;

}

2. GetBatch:

GetBatch Endpoint URL: https://prowess.cmie.com/api/getbatch

GetBatch example using Shell-Curl:

curl -F &quot;apikey=YOUR\_API\_KEY&quot; \

-F &quot;token=TOKEN&quot; \

-X POST \

https://prowess.cmie.com/api/getbatch -o -

GetBatch example using PHP:

&lt;?php

$ch= curl\_init();

curl\_setopt($ch, CURLOPT\_URL, &#39;https://prowess.cmie.com/api/getbatch&#39;);

curl\_setopt($ch, CURLOPT\_POST, true);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, [ &#39;apikey&#39; =&gt; &#39;YOUR\_API\_KEY&#39;, &#39;token&#39; =&gt;

&#39;TOKEN&#39; ]);

$response= curl\_exec($ch);

if(curl\_errno($ch))

printf(&quot;Error: %s\n&quot;, curl\_error($ch));

else

printf(&quot;Response: %s\n&quot;, $response);

curl\_close($ch);

?&gt;

GetBatch example using Python:

import requests

url= &#39;https://prowess.cmie.com/api/getbatch&#39;

zipfile= &#39;TOKEN.zip&#39;

mydata= { &#39;apikey&#39;: &#39;YOUR\_API\_KEY&#39;, &#39;token&#39;: &#39;TOKEN&#39; }

response= requests.post(url, data = mydata)

if response.status\_code == 200:

if response.headers[&quot;Content-Type&quot;] == &quot;application/json&quot;:

print(&quot;response:&quot;, response.content)

else:

with open(zipfile,&#39;wb&#39;) as output\_zip:

output\_zip.write(response.content)

else:

print(&#39;Failed to download file: {response.status\_code}&#39;)

GetBatch Response (JSON):

{

&quot;status&quot; : &quot;200&quot;,

&quot;message&quot; : &quot;IN\_QUEUE&quot;,

&quot;errcode&quot; : 0,

&quot;errdesc&quot; : &quot;Success&quot;,

&quot;servertime&quot; : &quot;01-03-24 17:52:02&quot;

}

NOTE: &quot;message&quot; can be:

1. IN\_QUEUE

2. PROCESSING

3. PROCESSED

4. ZIP WAITING TO BE QUEUED

5. ZIP INQUEUE

6. ZIP PROCESSING

Each &quot;message&quot; denotes the stage in which your current request is in the message order.

Once the &#39;.zip&#39; is ready it will be streamed directly upon request, without any status message.