RDP ESG CFS File Workflow

Importing libararies

In [2]:
```
import os
import sys
import requests
import json
from dotenv import dotenv_values
config = dotenv_values(".env")
```

Set RDP credentials and Initial Parameters

In [3]:
```
username = config['RDP_USERNAME'] #Or replace with your RDP Machin-ID
password = config['RDP_PASSWORD'] #Or replace with your RDP Password
clientId = config['RDP_APP_KEY'] #Or replace with your RDP APP Key

RDP_HOST= 'https://api.refinitiv.com'
acccess_token = None
refresh_token = None
expires_in = 0
```

RDP APIs Application Workflow

Step 1: Authentication with RDP APIs

Refinitiv Data Platform entitlement check is based on OAuth 2.0 specification. The first step of an application workflow is to get a token from RDP Auth Service, which will allow access to the protected resource, i.e. data REST API.

The API requires the following access credential information:

- Username: The username.
- Password: Password associated with the username.
- Client ID: This is also known as `AppKey`, and it is generated using an App key Generator. This unique identifier is defined for the user or application and is deemed

confidential (not shared between users). The client_id parameter can be passed in the request body or as an "Authorization" request header that is encoded as base64.

The HTTP request for the RDP APIs Authentication service is as follows:

```
POST /auth/oauth2/v1/token HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Host: api.refinitiv.com:443
Content-Length: XXX

username=RDP_USERNAME
&password=RDP_PASSWORD
&client_id=RDP_APP_KEY
&grant_type=password
&takeExclusiveSignOnControl=true
&scope=trapi
```

Once the authentication success, the function gets the RDP Auth service response message and keeps the following RDP token information in the variables.

- **access_token**: The token used to invoke REST data API calls as described above. The application must keep this credential for further RDP APIs requests.
- **refresh_token**: Refresh token to be used for obtaining an updated access token before expiration. The application must keep this credential for access token renewal.
- **expires_in**: Access token validity time in seconds.

Next, after the application received the Access Token (and authorization token) from RDP Auth Service, all subsequent REST API calls will use this token to get the data. Please find more detail regarding RDP APIs workflow in the following resources:

- RDP APIs: Introduction to the Request-Response API page.
- RDP APIs: Authorization - All about tokens page.

In [4]:
```
#step 1 - get RDP Access Token from RDP

# Send HTTP Request
auth_url = f'{RDP_HOST}/auth/oauth2/v1/token'
payload = f'grant_type=password&username={username}&client_id={clientId}&password={
try:
    response = requests.post(auth_url,
                             headers = {'Content-Type':'application/x-www-form-urle
                             data = payload,
                             auth = (clientId, '')
                )
except requests.exceptions.RequestException as exp:
    print(f'Caught exception: {exp}')


if response.status_code == 200:  # HTTP Status 'OK'
    print('Authentication success')
```

```
    access_token = response.json()['access_token']
    refresh_token = response.json()['refresh_token']
    expires_in = int(response.json()['expires_in'])

if response.status_code != 200:
    print(f'RDP authentication failure: {response.status_code} {response.reason}')
    print(f'Text: {response.text}')
```

```
Authentication success
```

Requesting Data from RDP APIs

That brings us to requesting the RDP APIs data. All subsequent REST API calls use the Access Token via the *Authorization* HTTP request message header as shown below to get the data.

- Header:
  - Authorization = `Bearer <RDP Access Token>`

Please notice *the space* between the `Bearer` and `RDP Access Token` values.

The application then creates a request message in a JSON message format or URL query parameter based on the interested service and sends it as an HTTP request message to the Service Endpoint. Developers can get RDP APIs the Service Endpoint, HTTP operations, and parameters from Refinitiv Data Platform's API Playground page - which is an interactive documentation site developers can access once they have a valid Refinitiv Data Platform account.

Requesting CFS Data

Step 2: Listing the Package Ids using the Bucket Name

Note: **If you already know your package Ids, you can skip to #step3**

To request the CFS data, the first step is to send an HTTP `GET` request to the RDP `/file-store/v1/packages?bucketName={bucket-name}` endpoint to list all Package Ids under the input `bucket-name` .

The HTTP Request structure is as follows:

```
GET /file-store/v1/packages?bucketName={bucket-name} HTTP/1.1
Host: api.refinitiv.com
Authorization: Bearer <Access Token>
```

Note: The bucket name is *case-insensitive*.

The ESG bucket name is **bulk-ESG**.

In [5]:
```
#step 2 - list Package IDs from bucket name
```

```
CFS_url = f'{RDP_HOST}/file-store/v1/packages?bucketName=bulk-ESG'

try:
    response = requests.get(CFS_url, headers={'Authorization': f'Bearer {access_tok
except requests.exceptions.RequestException as exp:
    print(f'Caught exception: {exp}')

if response.status_code == 200:  # HTTP Status 'OK'
    print('Receive list Package IDs from RDP APIs')
else:
    print(f'RDP APIs: CFS request failure: {response.status_code} {response.reason}
    print(f'Text: {response.text}')
```

```
Receive list Package IDs from RDP APIs
```

Example of the first entry of package IDs, the pacakgeId is the `packageId` field.

In [7]:
```
print(json.dumps(response.json()['value'][0], sort_keys=True, indent=2, separators=
```


```
{
  "bucketNames":[
    "bulk-ESG"
  ],
  "contactEmail":"robin.fielder@refinitiv.com",
  "created":"2021-11-11T07:54:04Z",
  "modified":"2023-02-10T09:10:16Z",
  "packageId":"4037-e79c-96b73648-a42a-6b65ef8ccbd1",
  "packageName":"Bulk-ESG-Global-Symbology-Organization-v1",
  "packageType":"bulk"
}
```


The next step is choosing the package Id.

In [8]:
```
packageId = response.json()['value'][0]['packageId']
packageId
```

Out[8]:
```
'4037-e79c-96b73648-a42a-6b65ef8ccbd1'
```

Step 3: Listing the Filesets of the Bulk ESG Data with the packageId

The next step is calling the CFS API with the buket name and package Id to list all FileSets using `bucket-name` and `packageId`.

API endpint is `/file-store/v1/file-sets?bucket=bulk-ESG&packageId={packageId}`

The HTTP Request structure is as follows:

```
GET /file-store/v1/file-sets?bucket=bulk-ESG&packageId={packageId}
HTTP/1.1
Host: api.refinitiv.com
Authorization: Bearer <Access Token>
```

In [24]:
```
#step 3 - get file id from bucket name

CFS_url = f'{RDP_HOST}/file-store/v1/file-sets?bucket=bulk-ESG&packageId={packageId

try:
    response = requests.get(CFS_url, headers={'Authorization': f'Bearer {access_tok
except requests.exceptions.RequestException as exp:
    print(f'Caught exception: {exp}')


if response.status_code == 200:  # HTTP Status 'OK'
    print('Receive FileSets list from RDP APIs')
else:
    print(f'RDP APIs: CFS request failure: {response.status_code} {response.reason}
    print(f'Text: {response.text}')
```

```
Receive FileSets list from RDP APIs
```

In [25]:
```
print(json.dumps(response.json()['value'][1], sort_keys=True, indent=2, separators=
```


```
{
  "attributes":[
    {
      "name":"ContentType",
      "value":"Symbology Organization"
    }
  ],
  "availableFrom":"2023-12-10T16:17:46Z",
  "availableTo":"2023-12-24T16:17:46Z",
  "bucketName":"bulk-ESG",
  "contentFrom":"1970-01-01T00:00:00Z",
  "contentTo":"2023-12-10T15:55:00Z",
  "created":"2023-12-10T16:17:46Z",
  "files":[
    "461e-dabc-eb88f134-a58e-ee517bf5f098"
  ],
  "id":"4aee-547a-a1fe9073-a35b-20a69cb47798",
  "modified":"2023-12-10T16:17:51Z",
  "name":"Bulk-ESG-Global-Symbology-Organization-v1-Jsonl-Init-2023-12-10T16:02:01.9
26Z",
  "numFiles":1,
  "packageId":"4037-e79c-96b73648-a42a-6b65ef8ccbd1",
  "status":"READY"
}
```


The File ID is in the `files` array

In [26]:
```
# try just one file
file_id = response.json()['value'][0]['files'][0]
file_id
```

Out[26]:
```
'4823-5fb2-67ae60ab-bd45-1699a214c428'
```

Step 3.1: Paging

By default, the `/file-store/v1/file-sets` and `/file-store/v1/packages?bucketName={bucket-name}` endpoints always return 25 results per request. You can adjust the number of return results via the `pageSize` query parameter, the maximum number is **100.**

The result also has the `@nextLink` node that contains the URL for requesting the next page of query.

You can find more detail about the Paging and @nextLink node on the step 3.1 of the A Step-By-Step Workflow Guide for RDP Client File Store (CFS) API article and GitHub.

Step 4: Get the file URL on AWS S3

The last step is downloading the FIle using File ID with the RDP `/file-store/v1/files/{file ID}/stream` endpoint.

The HTTP Request structure is as follows:

```
GET /file-store/v1/files/{fileId}/stream?doNotRedirect=true HTTP/1.1
Host: api.refinitiv.com
Authorization: Bearer <Access Token>
```

In [27]:
```
#step 3 - get file stream (content) from file id

FileID_url = f'{RDP_HOST}/file-store/v1/files/{file_id}/stream?doNotRedirect=true'

try:
    response = requests.get(FileID_url, headers={'Authorization': f'Bearer {access_
except requests.exceptions.RequestException as exp:
    print(f'Caught exception: {exp}')


if response.status_code == 200:  # HTTP Status 'OK'
    print('Receive File URL from RDP APIs')
else:
    print(f'RDP APIs: CFS request failure: {response.status_code} {response.reason}
    print(f'Text: {response.text}')
```

```
Receive File URL from RDP APIs
```

The File URL is in the `url` attribute.

In [28]:
```
file_url = response.json()['url']
file_url
```

Out[28]:
'https://a206464-bulk-esg.s3.amazonaws.com/Bulk-ESG-Global-Symbology-Organization-v1/2023/12/03/Bulk-ESG-Global-Symbology-Organization-v1-Init-2023-12-03T16%3A03%3A00.312Z.jsonl.gz?x-request-Id=79ec4301-3c90-410d-8669-cd8a3208817b&x-package-id=4037-e79c-96b73648-a42a-6b65ef8ccbd1&x-client-app-id=b4842f3904fb4a1fa18234796368799086c63541&x-file-name=Bulk-ESG-Global-Symbology-Organization-v1-Init-2023-12-03T16%3A03%3A00.312Z.jsonl.gz&x-fileset-id=4721-716d-682abf0a-86a7-2a54368ee8a7&x-bucket-name=bulk-ESG&x-uuid=GESG1-178570&x-file-Id=4823-5fb2-67ae60ab-bd45-1699a214c428&x-fileset-name=Bulk-ESG-Global-Symbology-Organization-v1-Jsonl-Init-2023-12-03T16%3A03%3A00.312Z&x-event-external-name=cfs-claimCheck-download&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEPj%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJHMEUCIE%2BhXciZwIPFjlPzDqmdF1nZG1dXHEKzkP%2F4owJlECInAiEA03cm4ciwSbS%2F0RK5Zqed2n2IsXs%2F71kiOs0ZI45PzXcqmgIIcBAEGgw2NDIxNTcxODEzMjYiDLg9eof1aLyab%2BOFJyr3AaYTRqezPAsS3hd9cu9%2BGSrxopxflq9SOAe3%2B1wki%2F3%2F%2FeIL%2BWw%2FPSTq0fa4yipxS6raR4C%2F1oYP7P8TRrA%2BvDtSSOtDC2a8KAaul58anXi%2BNT34mcVTPd3rMDp9u%2Fa4xfn5wbepDjquto03erezymQnyIVxrQvjnzEuZ0t4xGypya4HeK%2F8sa3O4j1eV2Plt3tkaT4oL63FpR2%2FmEhyUKRZlZNlzRpgErxSnAgokSSjJyKoklyFy1uHcProojf01i8ZN8ss%2Fs6oiz0LAMiCop6pWlx8kqFNq%2FsQPA1MQiXT%2B65CnF6xYFpGZhDCiF2MmRLwirXFEtEwyfvvqwY6nQH5erE7WdqeTRC7w3MfTqTc4sLVo4o6t8Rl8AGHJ6EBbmIQUdhiCTcpwYgboJLaOyJKrvtZCSjBMySXSK2aBLr0sygh9eDG8m5zG%2FyawWngLqtfdf%2BYLUAYID9fN0kF%2B%2FeYBxy9m2K8ZfTWSXYI4oGUMd%2Fk1zMV668O7BsoOcqxs%2Bw9GA9XTDCHdyVFOHWS%2BOgscOfd0ywtnZnFZL5p&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20231215T071833Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAZLA4M7GHKBDYCR5I%2F20231215%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=71818ad5428ee9d944832c20a437b932447df0734a7f939c7f6f64929ae5d775'

Step 5: Downloading the file

Based on the S3 `file_url` above, the actual file name is *Bulk-ESG-Global-Symbology-Organization-v1-Init-2023-12-03T16_03_00.312Z.jsonl*. So you need to replace the escape character `%3A` with `_` (underscore) character.

Note: If you cannot download the file, please wait for a while and then retry download the file from the URL. Please do not flush the download requests.

In [29]:
```python
#Downlaod file
import polling2

zipfilename = file_url.split("?")[0].split("/")[-1].replace("%3A","_")
print(f'Downloading File {zipfilename} ...')

def test_result(response):
    return response.status_code == 200

try:
    response = polling2.poll(lambda: requests.get(file_url),
                             step = 10,
                             poll_forever = True,
                             check_success= test_result)
except requests.exceptions.RequestException as exp:
    print(f'Caught exception: {exp}')

if response.status_code == 200:  # HTTP Status 'OK'
    print('Receive File Successfully')
    open(zipfilename, 'wb').write(response.content)
```

```
    print(f'{zipfilename} Saved')
else:
    print(f'RDP APIs: Request file failure: {response.status_code} {response.reason
    print(f'Text: {response.text}')
```

```
Downloading File Bulk-ESG-Global-Symbology-Organization-v1-Init-2023-12-03T16_03_00.
312Z.jsonl.gz ...
Receive File Successfully
Bulk-ESG-Global-Symbology-Organization-v1-Init-2023-12-03T16_03_00.312Z.jsonl.gz Sav
ed
```

And then unzip the file.

The normal ESG Bulk file type provides data for developers in **.gz** file.

In [30]:
```
#unzip file
import gzip
import shutil
try:
    unzipfilename = zipfilename.split('.gz')[0]
    print(f'Unzip to {unzipfilename} ...')
    with gzip.open(zipfilename, 'rb') as f_in:
        with open(unzipfilename, 'wb') as f_out:
            shutil.copyfileobj(f_in, f_out)
    print('Done')
except Exception as e:
    print('The error is: ',e)
```

```
Unzip to Bulk-ESG-Global-Symbology-Organization-v1-Init-2023-12-03T16_03_00.312Z.jso
nl ...
Done
```

Step 6: Refresh Token with RDP APIs

Before the session expires (based on the `expires_in` parameter, in seconds) , an application needs to send a Refresh Grant request message to RDP Authentication service to get a new access token before further request data from the platform.

The API requires the following access credential information:

- Refresh Token: The current Refresh Token value from the previous RDP Authentication call
- Client ID: This is also known as `AppKey` , and it is generated using an App key Generator. This unique identifier is defined for the user or application and is deemed confidential (not shared between users). The client_id parameter can be passed in the request body or as an "Authorization" request header that is encoded as base64.
- Grant Type `refresh_token` : This is for getting a new Access Token.

The HTTP request for the RDP APIs Authentication service is as follows:

```
POST /auth/oauth2/v1/token HTTP/1.1
Accept: */*
```

```
Content-Type: application/x-www-form-urlencoded
Host: api.refinitiv.com:443
Content-Length: XXX

refresh_token={current_refresh_token}
&grant_type=refresh_token
&client_id=RDP_APP_KEY
```

Once the authentication success, the function gets **access_token**, **refresh_token**, and **expires_in** from the RDP Auth service response message the same as the previous RDP Authentication call. An application must keep those value for the next Refresh Token call.

Caution: API Limit

The RDP Authentication service has the API limit described on the RDP APIs: Limitations and Guidelines for the RDP Authentication Service article. If the application flushes the authentication request messages (both `password` and `refresh_token` grant_type) beyond the limit, the account will be blocked by the API Gateway.

In [35]:
```
#step 6 - Refreshing Token

# Send HTTP Request
auth_url = f'{RDP_HOST}/auth/oauth2/v1/token'
payload = f'grant_type=refresh_token&client_id={clientId}&refresh_token={refresh_to
try:
    response = requests.post(auth_url,
                             headers = {'Content-Type':'application/x-www-form-urle
                             data = payload,
                             auth = (clientId, '')
                )
except requests.exceptions.RequestException as exp:
    print(f'Caught exception: {exp}')

if response.status_code == 200:  # HTTP Status 'OK'
    print('Refresh Token success')
    access_token = response.json()['access_token']
    refresh_token = response.json()['refresh_token']
    expires_in = int(response.json()['expires_in'])

if response.status_code != 200:
    print(f'RDP authentication failure: {response.status_code} {response.reason}')
    print(f'Text: {response.text}')
```

```
Refresh Token success
```

Step 7: Revoke Token to ending the session.

This revocation mechanism allows an application to invalidate its tokens if the end-user logs out, changes identity, or exits the respective application. Notifying the authorization server that the token is no longer needed allows the authorization server to clean up data associated with that token (e.g., session data) and the underlying authorization grant.

The API requires the following HTTP Header and Credential parameter information:

- Header:
  - Authorization = `Basic <App Key in Base64 format>`

Please notice *the space* between the `Basic` and `App Key in Base64 format` values.

- Body parameter
  - token: The current `Access Token` value from the previous RDP Authentication call

The HTTP request for the RDP APIs Authentication service is as follows:

```
POST /auth/oauth2/v1/revoke HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
Host: api.refinitiv.com:443
Authorization: Basic <App Key in Base64>
Content-Length: XXX

token={current_Access_token}
```

In [36]:
```
#step 7 - Revoking Token

import base64

clientId_bytes = clientId.encode('ascii')
base64_bytes = base64.b64encode(clientId_bytes)
clientId_base64 = base64_bytes.decode('ascii')

# Send HTTP Request
auth_url = f'{RDP_HOST}/auth/oauth2/v1/revoke'
payload = f'token={access_token}'
try:
    response = requests.post(auth_url,
                             headers = {
                                 'Content-Type':'application/x-www-form-urlencoded'
                                 'Authorization': f'Basic {clientId_base64}'
                             },
                             data = payload,
                             auth = (clientId, '')
                )
except requests.exceptions.RequestException as exp:
    print(f'Caught exception: {exp}')

if response.status_code == 200:  # HTTP Status 'OK'
    print('Revoke Token success')
if response.status_code != 200:
    print(f'RDP authentication failure: {response.status_code} {response.reason}')
    print(f'Text: {response.text}')
```

```
Revoke Token success
```

References

That brings me to the end of my ESG CFS API workflow project. For further details, please check out the following resources:

- Refinitiv Data Platform APIs page on the Refinitiv Developer Community website.
- Refinitiv Data Platform APIs Playground page.
- Refinitiv Data Platform APIs: Introduction to the Request-Response API.
- Refinitiv Data Platform APIs: Authorization - All about tokens.
- Limitations and Guidelines for the RDP Authentication Service article.
- Getting Started with Refinitiv Data Platform article.
- ESG Data Guide
- ESG-Bulk CFS API User Guide
- ESG Bulk - Point in Time User Guide

For any questions related to Refinitiv Data Platform APIs, please use the RDP APIs Forum on the Developers Community Q&A page.

In [ ]: