TM3 CFS File Workflow

Last Updated: Aug 2025

Importing libararies

In [1]:
```
import os
import sys
import requests
import json
from dotenv import dotenv_values
config = dotenv_values('.env')
```

Set RDP credentials and Initial Parameters

In [2]:
```
username = config['MACHINE_ID'] #Or replace with your RDP Machine-ID
password = config['PASSWORD'] #Or replace with your RDP Password
clientId = config['APP_KEY'] #Or replace with your RDP APP Key

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

Step 1: Authentication with RDP APIs

RDP APIs 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.

- Grant Type `password` : This is for initial authentication request. An application does not have any token, so it requests new tokens using username/password combination.

```
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
```

In [23]:
```
#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:
    auth_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 auth_response.status_code == 200:  # HTTP Status 'OK'
    print('Authentication success')
    access_token = auth_response.json()['access_token']
    refresh_token = auth_response.json()['refresh_token']
    expires_in = int(auth_response.json()['expires_in'])

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

```
Authentication success
```

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.

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 via the *Authorization* HTTP request message header as shown below.

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

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

Step 2: Query for specific Date using modifiedSince and pageSize=100 to limit data return<-- Recommended One

Now we come to getting the `FileSets` information. The `modifiedSince` parameter can help an application to limit the returned File-Set only for the File-Set that has been modified after a specified time.

The application needs to send an HTTP `GET` request to the RDP `/file-store/v1/file-sets?bucket=buck-Custom&packageId={packageId}&pageSize=100&modifiedSince={datetime}` endpoint to list **the first** 100 FileSets under the input `bucket-name` and `packageId` that has been modified since `modifiedSince` as follows.

```
GET /file-store/v1/file-sets?bucket={bucket-name}&packageId=
{packageId}&modifiedSince={datetime}&pageSize=100 HTTP/1.1
Host: api.refinitiv.com
Authorization: Bearer <Access Token>
```

Example:

```
GET /file-store/v1/file-sets?bucket=bulk-Custom&packageId=xxxx-bbbb-yyy-
aaaa&pageSize=100&modifiedSince=2025-08-20T00:00:00Z HTTP/1.1
Host: api.refinitiv.com
Authorization: Bearer <Access Token>
```

The `bucket-name` for the TM3 (Municipal Market Monitor) feed data is **bulk-Custom**.

Please contact your LSEG representative to help you with the `packageId`.

In [4]:
```
# pick the packageId you need and set to the packageId variable
#packageId = response.json()['value'][0]['packageId']
packageId = config['PACKAGE_ID']
bucket_name = 'bulk-Custom'
```

In [24]:
```
#step 2 - list FileSets from bucket name and package Id and modifiedSince

CFS_url = f'{RDP_HOST}/file-store/v1/file-sets'

params = {
    'bucket': bucket_name,
    'packageId': packageId,
    'pageSize': 100,
    'modifiedSince':'2025-07-25T12:00:00Z'
}
```

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

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

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

Example of FileSets

In [ ]:
```
print(json.dumps(fileSet_response.json()['value'], sort_keys=True, indent=2, separa
```

The File ID is in the **files** array. Select the one that you need.

I am demonstrating with the first entry.

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

Step 2.5: Or you can listing the 100 records of the FileSets using the Bucket Name and Package ID and pageSize=100

```
GET /file-store/v1/file-sets?bucket={bucket-name}&packageId=
{packageId}&pageSize=100 HTTP/1.1
Host: api.refinitiv.com
Authorization: Bearer <Access Token>
```

In [14]:
```
#step 2.5 - list FileSets from bucket name and package Id

CFS_url = f'{RDP_HOST}/file-store/v1/file-sets'

fileSet_response = None

params = {
    'bucket': bucket_name,
    'packageId': packageId,
    'pageSize': 100
}

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

if fileSet_response.status_code == 200:  # HTTP Status 'OK'
    print('Receive FileSets list from RDP APIs')
else:
```

```
    print(f'RDP APIs: CFS request failure: {fileSet_response.status_code} {fileSet_
    print(f'Text: {fileSet_response.text}')
```

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

Example of theFileSets.

In [ ]:
```
print(json.dumps(fileSet_response.json()['value'], sort_keys=True, indent=2, separa
```

The FileSets can be more than 100!!

Please note that the FileSets can be more than 100 records. The API returns data maximum 100 records per one query.

If the there are more than 100 records, the API returns the `@nextLink` node which contains the URL for requesting the next page of query as follows:

```
GET {@nextLink URL}, HTTP/1.1
Host: api.refinitiv.com
Authorization: Bearer <Access Token>
```

In [16]:
```
if '@nextLink' in fileSet_response.json():
    next_link = fileSet_response.json()['@nextLink']
    print(next_link)
```

```
/file-store/v1/file-sets?bucket=bulk-Custom&packageId=4a2f-a444-b1f4a145-b0f9-4509ed
361868&pageSize=100&skipToken=ZmlsZXNldElkPTQwZDQtNDIwMC03NmJlNWE3NS05OTVmLWU0NGM0OT
U1MTM4MQ
```

You can use `@nextLink` to the URL of the HTTP Request GET Method.

In [17]:
```
if '@nextLink' in fileSet_response.json():
    next_link = fileSet_response.json()['@nextLink']
    CFS_url = f'{RDP_HOST}{next_link}'

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

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

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

In [ ]:
```
print(json.dumps(fileSet_response.json(), sort_keys=True, indent=2, separators=(','
```

Then you can continue to send requests to URL in `@nextLink` node to get the next page results.

More on /file-store/v1/file-sets?bucket parameters

Beside the `bucket` and `packageId` queries, the `/file-store/v1/file-sets?bucket` endpoint supports the following optional parameters:

- *name*: The name of the file-set. Only exactly matched results are returned.
- *packageId*: Package ID
- *status*: Filter file-set by status (Ready/Pending)
- *availableFrom*: Return all file-sets that become visible to permissioned users after the specified Datetime.
- *availableTo*: Return all file-sets that is no longer visible to permissioned user after the specified Datetime.
- *contentFrom*: Filter results by the age of the content within the file-set.
- *contentTo*: Filter results by the age of the content within the file-set.
- *createdSince*: Return all file-sets that have a created date after the specified Datetime
- *modifiedSince*: Return all file-sets that have a modified date after the specified Datetime.
- *attributes*: Return a list of publisher-defined attributes of the file-sets.
- *pageSize*: The number of file-sets that will be shown on one page. Default value is 25.
- *skipToken*: A token to retrieve the next set of file-set result that exceeds page size.

Please find more detail on the CFS API User Guide document.

Step 3: Get the AWS S3 file URL using FileSet

The next step is getting the file URL on Amazon AWS S3 service with the RDP `/file-store/v1/files/{fileSet}/stream` endpoint.

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

In [19]:
```
#step 3 - get file URL from file id

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

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

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

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

The File URL is in the `url` attribute of the response message.

In [ ]:
```
file_url = fileID_response.json()['url']
print(file_url)
```

More on /file-store/v1/files/ parameters

Beside the `file_id` query, the `/file-store/v1/files/` endpoint supports the following optional parameters:

- *createdSince*: Return all files that have a created date after the specified Datetime.
- *modifiedSince*: Return all files that have a modified date after the specified Datetime.
- *pageSize*: The number of files that will be shown on one page. Default value is 25.
- *skipToken*: A token to retrieve the next set of file result that exceeds page size

Please find more detail on the CFS API User Guide document.

Step 4: Downloading the actual file from AWS

Once you got the S3 URL. You can download the bulk file using that URL (**as is**). **Do not alter or make any changes to the URL text string**. It will cause unable to download or signature mismatch error.

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.
- The code below set `verify = False` property in a `requests` library call to workaround LSEG's beloved ZScaler blocks a download request message. **Do not** set `verify = False` in a Production.

In [ ]:
```
#step 4 - Download file
import polling2

try:
    print(f'Downloading File from {file_url} ...')
    bulkFile_response = polling2.poll(lambda: requests.get(file_url, verify= False)
                            step = 10,
                            poll_forever = True,
                            check_success= lambda r: r.status_code == 200)
except requests.exceptions.RequestException as exp:
    print(f'Caught exception: {exp}')

# Save the file locally.
if bulkFile_response.status_code == 200:  # HTTP Status 'OK'
    zipfilename = file_url.split("?")[0].split("/")[-1].replace("%3A","_")
    print('Download File Successfully')
    open(zipfilename, 'wb').write(bulkFile_response.content)
    print(f'{zipfilename} Saved')
else:
```

```
    print(f'RDP APIs: Request file failure: {bulkFile_response.status_code} {bulkFi
    print(f'Text: {bulkFile_response.text}')
```

Now you get the CFS file that you can extract and read the file.

Step 5: 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.

```
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
```

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 [10]:
```
#step 5 - 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
auth_response = None

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

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

```
Refresh Token success
```

Step 6: 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>`
- Body parameter
  - token: The current `Access Token` value from the previous RDP Authentication call

Please be noticed

- The **":"** string after the **App Key** as the RDP does not use the password for this Basic Authentication.
- The `/revoke` endpoint does not use the `password` in a Basic Authorization, so we need to send empty password in `AppKey:` format.

```
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}
```

I am demonstrating with the Python requests library. We can use the `auth(clientid,'')` to the `requests.post` function, and the library handles the base64 conversion for us.

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

# Send HTTP Request
auth_url = f'{RDP_HOST}/auth/oauth2/v1/revoke'
payload = f'token={access_token}'
auth_response = None

try:
    auth_response = requests.post(auth_url,
                                  headers = {
                                        'Content-Type':'application/x-www-form-urlenc
                                        },
                                  data = payload,
                                  auth = (clientId, '')
                )
except requests.exceptions.RequestException as exp:
    print(f'Caught exception: {exp}')

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

```
Revoke Token success
```

In [ ]:
```

```