NTT DATA Business Solutions A/S • Erhvervsbyvej 11 • DK-8700 Horsens

Tlf.: +45 7022 2166
info-solutions-dk@nttdata.com

www.nttdata-solutions.com/dk

External storage API documentation

Date 08/09-2023

1 Contents

2 Overview of methods

The intended usage for this API is for customers to upload files to the Momenttum platform. The API has the following methods:

Method	Intended usage
PUT (Upload file)	Creates or replaces a file in the storage container
GET (List)	Get a list of files currently in the storage container
GET (File)	Get a specific file from the storage container.

3 Configuration

The use of the API requires multiple configuration parameters. These parameters are all provided by NTTData to allow usage of the API

Name	Description
API-key	The key required for the specific API
Client_ID	The Client_ID provided by NTTData for authentication
Client_Secret	The Client_secret provided by NTTData for authentication
Tenant_ID	Tenant_ID where the API is located for authentication
Scope	The scope containing the API to limit the OAuth2 token
API_URL	The base URL of the API

4 Security

The usage of the API requires both OAuth2 and an API key to ensure security.

4.1 OAuth2 Authentication with MSAL

OAuth2 authentication is done with the Microsoft authentication service and uses **the Client_ID, Client_Secret, Scope** and **Tenant_ID.** The authentication will provide an **access_token** that is to be used in follow up api calls. The Microsoft documentation for this can be found at https://learn.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-auth-code-flow

An example of how an OAuth2 token can be retrieved with the MSAL python library can be seen below:

```python
from msal import ConfidentialClientApplication

def get_token_with_msal(client_id, client_secret, auth_url, scope):
    try:
        app = ConfidentialClientApplication(
            client_id=client_id,
            authority=auth_url,
            client_credential=client_secret
        )

        result = app.acquire_token_silent(scope, account=None)

        if not result:

            result = app.acquire_token_for_client(scopes=scope)

        if "access_token" in result:
            return result['access_token']
        else:
            print(result.get("error"))
            print(result.get("error_description"))
            print(result.get("correlation_id"))
            return None

    except Exception as e:
        print(f"An error occurred: {str(e)}")
        return None
```

4.2 Authorizing an API call

Any call to any API endpoint requires two specific headers. The Authorization and the api-key header. These authorize the API call.

Header name	Content
Authorization	"bearer {token}"
Api-key	"{api_key}"

5 Specific API information

More in depth information on the specific APIs with a python example for each type of API call

5.1 PUT (file)

The put method requires an additional header for the final file name on the storage account along with the data that has to be sent. Sample code for this operation can be seen below:

```python
import requests

def put_file(api_url, token, api_key, intended_filename, file_path):
    headers = {"Authorization": f"Bearer {token}",
            "api-key": api_key,
            "Filename": intended_filename}

    try:
        with open(file_path, 'rb') as file:
            file_content = file.read()
            response = requests.put(api_url, headers=headers, data=file_content)


            if response.status_code == 201:
                print(f"Uploaded {file_path} as {intended_filename} successfully.")
            else:
                print(f"File upload failed.")

            return response
    except Exception as e:
        print(f"An error occurred: {str(e)}")
```

5.2 GET (list)

The get list operation requires that a parameter is set in the URL. Specifically, the API-URL should end with **"/?method=list".** In the python example this is handled by the requests library using the **params** parameter in the get request.

```python
import requests

def get_list(api_url, token, api_key):

    payload = {'method': 'list'}

    headers = {"Authorization": f"Bearer {token}",
               "api-key": api_key}

    response = requests.get(api_url, headers = headers, params = payload)

    if response.status_code == 200:
        print(f"List of files retrieved successfully")
    else:
        print(f"Get list failed.")

    return response
```

5.3 GET (file)

The get request for a file uses a single extra header in the form of the **"Filename"** header to fetch a specific file.

```
import requests

def get_file(api_url, token, api_key, filename):

    headers = {"Authorization": f"Bearer {token}",
               "api-key": api_key,
               "filename": filename}

    response = requests.get(api_url, headers = headers)

    if response.status_code == 200:
        print(f"List of files retrieved succesfully")
    else:
        print(f"Get list failed.")

    return response
```

6 Response codes

Here is a small list of possible response codes and their meaning

Code	Meaning
200	Success (ok)
201	Created successfully – the PUT operation was a success
401	Unauthorized - Some part of the authentication failed. A more precise description is usually provided as an HTML response