Brief

Return a tree representation of a Microsoft Graph Folder structure, returning all IDs, names, folders and files. For each folder, validate the name of the folder against a provided configuration. If all levels are valid, upload result to the S3 bucket with the ID of the file.

System must be written in Node.js.

See https://docs.microsoft.com/en-us/graph/api/resources/onedrive?view=graph-rest-1.0

Must use the npm package aws-sdk

Must use superagent

```
return request
.post(`${GRAPH_BASEURL}/me/drive`)
.set('Authorization', 'Bearer ' + token)
.send({
})
.then(response => {
return response.body
});
```

Start

The node process will have the following options

Variable	Description
token	A token that can be used in Microsoft graph
start_folder_id	ID of a folder from the graph
configuration	Configuration of how the directory structure should be validated. When run I should be able to change the levels. More values to be added as needed. let configuration: { config: [{level:0, type: "client"}, {level:1, type: "group"} {level:2, type: "cabinet"}]}
AWS_ACCESS_KEY_ID	Access Key ID for S3
AWS_SECRET_ACCESS_KEY	Access secret for S3
Bucket	Name of the bucket

Step 1: Load From MS Graph

Using the provided graph token, return all folders as a tree starting from the start_folder_id, including its name, id, folders and files (file should include size).

Result should look as follows

[

{id:1, name:'dir1', folders:[{id:11, name: 'sub dir1', folders:[], files:[{id:22, name: 'file name', size: 100}]}], files:[...]},

{id:2, name:'dir2, folders: [...], files:[...]}

}

Step 2: Validate Data

Assuming you have the following functions (return a promise, the resut is an object)

getClient(name) returns an object or null

getGroup(name) returns an object or null

getCabinet(name) returns an object or null

For each folder in the tree result, check if the name exists for the corresponding type based on the configuration.

Using the example result 'dir 1' is on level 0 of the tree structure, so it should use getClient('dir 1').

Using the example result 'sub dir 1' is on level 1 of the tree structure, so it should use getGroup('sub dir 1').

Set the result of the function to a property on the folder the result from the function 'value=result'

let r = getClient('dir 1)

{id:1, name:'dir1', value:r}

Upload to S3

Recursively going through the tree result, if all levels preceding and including the current folder have a value, we want all the files for that folder to be uploaded to S3

1. Download the file from graph
2. Upload file to S3 bucket using the id of the file
3. If an error is return, set a property on the row with the error
    - {error: err}