Modules

As of *10/06/2020*, we have 4 modules built:

Download a file

-
  - From eFileCabinet to another source
  - Uses connection "eFileCabinet oauth2"

Required Inputs Using Mappable Parameters

- File
  - A way for the user to dig into the file structure to find a valid file. This uses the RPC [List Files]

Interface

Note that this tab is only used when you need to specify to other modules what your output will be

- Data
  - This is the content of the data (file)that is going to be sent
- File Name
  - This is the name of the data (file) that is going to be sent
- File Size
  - This is the size of the data (file)that is going to be sent

Communication

Each number in the following ordered list is referring to a block in code

1. This block grabs the parent workspace id and initializes a counter so that the module will know where it currently is when it iterates in block 2. It also determines the last string in the path so that block 2 knows when it is done iterating
2. This block iterates through each string in the path to get the nodeId by it's name
   - It knows it is done when the `lastNamePath` is equal to its `currentName`
   - This `"condition": "{{if(parameters.path !== '/', true, false)}}"` will prevent it from running when there is no path entered so as to save an API call
3. This block uses the pathId and the accessToken to makes a call to our API so that we can GET the file to download

Upload a file

- From another source to eFileCabinet
- Uses connection "eFileCabinet oauth2"

Required Inputs Using Mappable Parameters

- Directory
  - This is where the user selects the Folder or Drawer that the file is to be saved in
  - This uses RPC List Folders
  - Notice that the name is called path, this is true because the value being return from the RPC is in format `abc/def/ghi`
- File Name
  - This will auto fill if the user "Source File" - (Other module input)
- Data
  - The file that will be uploaded and should be auto filled when another module links to it

Communication

Each number in the following ordered list is referring to a block in code

1. This block grabs the parent workspace id and initializes a counter so that the module will know where it currently is when it iterates in block 2. It also determines the last string in the path so that block 2 knows when it is done iterating
2. This block iterates through each string in the path to get the nodeId by it's name
   - It knows it is done when the `lastNamePath` is equal to its `currentName`
   - This `"condition": "{{if(parameters.path !== '/', true, false)}}"` will prevent it from running when there is no path entered so as to save an API call
3. This block processes the node that the user has input by selecting the last node. It then saves this id into a temp variable called "pathId"
4. This block attempts to create the file in the requested parent node
   - If there is an error (as seen in the response body), then it will take the suggestedName in the Batch Object name for block 3
5. *Conditional block:* This block will only run if the suggested name received in block 2 does not equal the file name parameter received
   - If this runs it will create the node with the suggested name
6. Now that the node has been created it will begin the upload process. This block will FileUpload POST to retrieve the `uploadIdentifier`
7. With the `uploadIdentifier` retrieved, this block will start the data upload
8. Once the data has finished uploading, it will finalize by sending `"complete": true`

Create a node

- - Create a new Folder, Drawer, or Cabinet
  - Uses connection "eFileCabinet oauth2"

Required Inputs Using Mappable Parameters

- Name
  - The name of the new node
- Node
  - A way for the user to select the parent Cabinet, Drawer, or Folder
  - This uses a mappable for of input, where the user just clicks the node in question, starting with the workspace and working down until they've selected the parent node in question
    - This uses the RPC List Folders

Communication

Each number in the following ordered list is referring to a block in code

1. This block grabs the parent workspace id and initializes a counter so that the module will know where it currently is when it iterates in block 2. It also determines the last string in the path so that block 2 knows when it is done iterating
2. This block iterates through each string in the path to get the nodeId by it's name
   - It knows it is done when the `lastNamePath` is equal to its `currentName`
   - This `"condition": "{{if(parameters.path !== '/', true, false)}}"` will prevent it from running when there is no path entered so as to save an API call
3. This block uses the pathid found in block 2 to make a call to our API so that we can fetch the systemType. It then saves that in to temp variable "parentSystemType".
4. This block starts by figuring the systemType of the new node, then it will POST the new node using the pathId, systemType, and name
   - If the parent is a workspace, it will create a cabinet
   - If the parent is a cabinet, it will create a drawer
   - If the parent is a drawer, it will create a folder
   - If the parent is a folder, it will create a folder

Rename a node

- - Rename a File, Folder, Drawer, or Cabinet
  - Uses connection "eFileCabinet oauth2"

Required Inputs Using Mappable Parameters

- This begins with a Select dropdown, where the user can select either a file or a folder
  - File - calls RPC List Files
  - Folder - calls RPC List Folders
- Name
  - The name they want the new item to be called

Communication

Each number in the following ordered list is referring to a block in code

1. This block grabs the parent workspace id and initializes a counter so that the module will know where it currently is when it iterates in block 2. It also determines the last string in the path so that block 2 knows when it is done iterating
2. This block iterates through each string in the path to get the nodeId by it's name
   - It knows it is done when the `lastNamePath` is equal to its `currentName`
   - This `"condition": "{{if(parameters.path !== '/', true, false)}}"` will prevent it from running when there is no path entered so as to save an API call
3. This block uses the pathId found in block 2 to update the node using PUT with the gathered name in the body

---

Revision #3
Created 18 July 2022 23:31:20 by Bryce Holloway
Updated 9 February 2023 20:20:50 by Bryce Holloway