Integromat

Getting Started

Setup

1. Head to integromat.com
2. Login
    - Username: development@efilecabinet.com
    - Password: Ask Trevor

eFileCabinet App

Click here to see our app.

An integromat app is composed of several components. The important ones are:

- Base
  - This is the parent structure that all modules and remote procedures inherits from, if you notice that every module is using a header or requiring some other piece of information, then place it here
- Connections
  - It is possible to create more than a single connection if a module requires something unique
  - The active connection is the one where oauth2 does it's magic
  - The current connection is setup in a way that it fetches the token and saves it inside of memory.
    - To access this token, call `connection.accessToken`
  - The "Common Data" section at the bottom allows us to hold parameters that we need, but wish to keep private such as the base64 encoded access string
    - This will need to be adjusted after we setup an integromat specific connection
  - Currently it is set up to renew the connection when there is less than one minute remaining
    - See: `"condition": "{{if(data.accessToken, data.expires < addMinutes(now, 1), true)}}"`
  - The "Parameters" tab is what indicates the user input. In this case, the user's username and password that will be used during authentication
- Modules
  - Modules are generally set up in a way to accomplish a single task, not single calls to our api
  - The "Communication" tab is where it will make it's calls to our api, save and manipulate temporary data, and perform basic logic
    - Any logic performed must follow the IML pattern see IML functions in the integromat documentation
  - The "Static Parameters" tab allows for user input only and is rarely used
    - To use parameters (static or mappable), simply call `parameter.ParameterName`
  - The "Mappable Parameters" tab allows for user input in a more fluid way. It also allows for Remote Procedures (RPC) to be called, allowing a better user experience.

This is what should generally always be used
    - I've noticed that it is easier to start here during module creation as you'll probably need to use or create a RPC to use or fetch additional input
    - To use parameters (static or mappable), simply call `parameter.ParameterName`

- # Remote Procedures (RPC)
    - Are sub modules that can are ran while the user is selecting parameters, before the module itself is ran
    - They are used to fetch input needed to run the app's module

- # IML Functions
    - IML is a feature that allows you to write your own JavaScript functions and execute them inside IML expressions to process data
    - IML functions are disabled by default. Luckily for you, I've already contacted them to unlock it.

Remote Procedure Calls (RPCs)

What are they?

- Think of RPCs as sub modules that are called to help the main modules get the inputs they need
- They CAN NOT be called inside the module itself, instead, they are called inside of the "Mappable Parameters" tab

How do I use them?

- List Files
  - Call this using `rpc://listFolders`
- List Folders
  - Call this using `rpc://listFiles`

List Files

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 RPC 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 will show the user the next layer in the path

Notice

```
"response": {
    "iterate": "{{body}}",
    "output": {
        "label": "{{item.name}}",
        "value": "{{item.id}}",
        "file": "{{if(item.systemType === 7, true, false)}}"
    }
}
```

- Our API responds with an array instead of an object, as a result Integromat does not know how to output the data
  - This means that we need to iterate through each element in the array and output the information we require
  - This `"file": "{{if(item.systemType === 7, true, false)}}"` is what determines if a proper file has been selected. If it evaluates to false then the module cannot proceed and will run the RPC again until it returns true

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 RPC 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 will show the user the next layer in the path

Notice

```
"response": {
    "iterate": "{{filterFolders(body)}}",
    "output": {
        "label": "{{item.name}}",
```

```
        "value": "{{item.id}}"
    }
}
```

- Our API responds with an array instead of an object, as a result Integromat does not know how to output the data
  - This means that we need to iterate through each element in the array and output the information we require
  - An IML function is called on the body to filter out all file node objects from the body

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