



Pipefy

Readme

Version 1.0

10/05/2022

Table of Contents

Readme	1
Introduction	3
Overview	3
Use cases	4
Requirements & Prerequisites	5
System Requirements	5
Prerequisites	5
Getting Started	6
Quick Start	6
Setup	6
Configuration and Use	7
How to get Pipefy's IDs (Fields, Pipes, Cards, etc)	8

1. Introduction

This document contains all essential information for the user to make full use of this A2019 [Pipefy Package](#). It includes a description of the functions and capabilities and step-by-step procedures for setup & configuration of the [Pipefy Package](#).

1.1 Overview

[Pipefy](#) is the workflow management software that makes business processes – such as purchasing, onboarding and recruiting – uncomplicated, so applicants, processors and managers are more efficient. Through automated workflows, Pipefy increases speed, increases visibility and delivers higher quality results without the need for IT implementation.

By combining Pipefy workflow automation with the Automation Anywhere platform, businesses can fully empower users to deploy digital workflows quickly, automating manual work across all business areas.

This package uses Pipefy's API to perform the most common actions available in the platform like Create/Get/Update cards or records.

These actions will require basic understanding of Pipefy's API to acquire the field IDs to use as the input of the actions, here's the example of a query to extract fields IDs from a Pipe and a Database, that you can then use as a input for any of the actions in the package:

```
{pipe(id: 123456){
  start_form_fields{
    id
    label
    type
    description
    options
  }
  phases{
    id
    name
    done
    fields{
      id
      label
      type
      description
      options
    }
  }
}}
```

```
{
  table(id: "ZtEdWh") {
    id
    name
    table_fields{
      id
      label
      type
      options
      description
    }
  }
}
```

All actions will also give you structured outputs always in the JSON format, as the example below:

```
{
  "data": {
    "action": "card.create",
    "card": { "id": 12345, "pipe_id": "CxeXHeOR" }
  }
}
```

Please refer to Pipefy's developers page for more: <https://developers.pipefy.com>

1.2 Use cases

The key use cases include:

- Automate procurement, vendor relationship and HR processes by integrating Pipefy to ERPs.
- Adding information from/to local excel spreadsheets.
- Leverage Automation Anywhere OCR and IDP capabilities by automating document driven workflows with Pipefy
- Import/export information on Pipefy with ease.

2. Requirements & Prerequisites

2.1 System Requirements

[Enterprise A2019 \(Cloud deployed\) and Community Edition device requirements.](#)

Review the machine hardware specifications, operating system versions, and browser types supported by Automation Anywhere Enterprise for creating and running bots and command packages as an Enterprise A2019 (Cloud deployed) or Community Edition user on your local machine.

2.2 Prerequisites

This package will use Pipefy's API to perform the actions, you need to have a Pipefy account with available API calls.

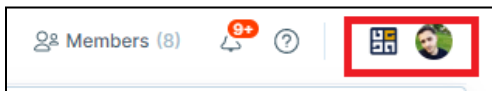
3. Getting Started

3.1 Quick Start

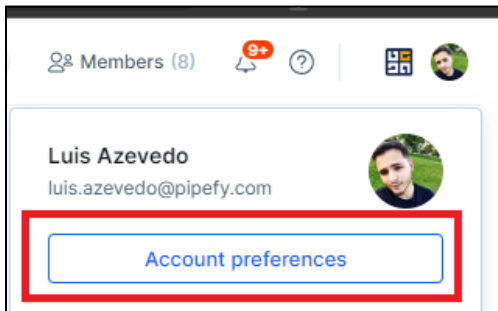
3.1.1 Setup

To start using the Pipefy Package you will need to first generate a token in your Pipefy account. You can follow the Step-by-Step instructions below to generate a token:

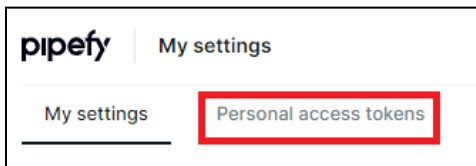
- After you log into Pipefy, click on the button on the top right of the screen that has your avatar and company logo



- Then you must click on "Account Preferences"



- Now click on the "Personal access tokens" tab in the newly opened window



- Click on "Generate new token"
- Inside the "Token description" insert the name of your token
- When you click on save, the token will be generated and then you can copy it for your Automation Anywhere Account

To use it inside our package, you must create a locker and store the newly created token as a credential (you can do that by following [this link](#)).

Now to use it inside a bot using an action from our package, you have to select the credential from that locker just like that:

Pipefy: Create Card

Creates a card for a Pipe based on the dictionary inserted and the Pipe's ID

Session Name

Pipefy Token

Credential
 Variable
 Insecure string

Locker - Lugui, Pipefy Lugui, Token
 ×

Now you are ready to use the Pipefy Package!

3.1.2 Configuration and Use

Action	Input Variables	Output
Create Card	<ul style="list-style-type: none"> Pipe ID Pipefy Token Session Name Dictionary with "Id of the fields" and their "respected values" 	Id of the card
Create Record	<ul style="list-style-type: none"> Table ID Pipefy Token Session Name Dictionary with "Id of the fields" and their "respected values" 	Id of the record
Custom Query	<ul style="list-style-type: none"> Pipefy Token Session Name Query 	JSON response
Upload Attachment	<ul style="list-style-type: none"> Pipefy Token Session Name File Name File Card ID Field ID 	JSON response
Move Card	<ul style="list-style-type: none"> Pipefy Token Session Name Card ID Destination Phase ID 	JSON response
Get Phase Cards	<ul style="list-style-type: none"> Pipefy Token Session Name Phase ID 	JSON response
Update Fields Values	<ul style="list-style-type: none"> Card ID Pipefy Token Session Name Dictionary with "Id of the fields" and their "respected values" 	JSON response
Get a Card by ID	<ul style="list-style-type: none"> Pipefy Token Session Name 	JSON response

	<ul style="list-style-type: none"> • Card ID 	
Get a Record By ID	<ul style="list-style-type: none"> • Pipefy Token • Session Name • Record ID 	JSON response
Delete Card	<ul style="list-style-type: none"> • Pipefy Token • Session Name • Card ID 	JSON response

To see an example of each JSON response [click here](#).

3.1.3 How to get Pipefy's IDs (Fields, Pipes, Cards, etc)

Access Pipefy's GraphQL Explorer (<https://app.pipefy.com/graphiql>)

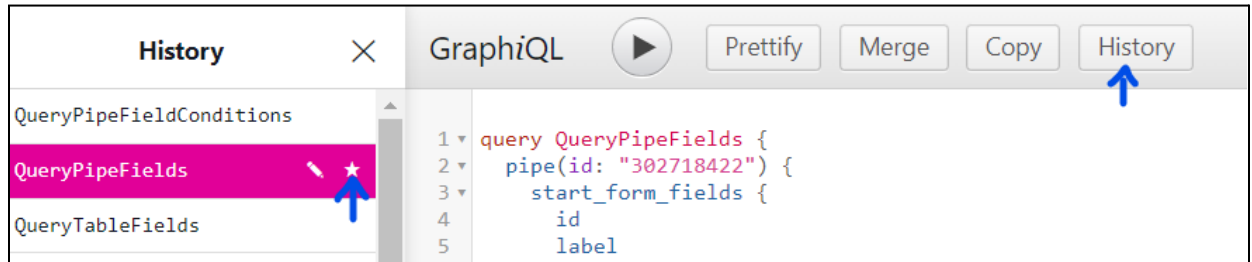
NOTE: You must be logged into your Pipefy account, this way you will be automatically authenticated when accessing the GraphQL Explorer.


On the left side of the screen, where we insert the queries/mutations, paste the following query replacing "<pipe_id>" in the 2nd line by the id of the pipe you want to query (Eg: "pipe(id: "12345") {"):

```
query QueryPipeFields {
  pipe(id: "<pipe_id>") {
    start_form_fields {
      label
      type
      id
      internal_id
    }
    phases {
      name
      id
      fields {
        label
        type
        id
        internal_id
      }
    }
  }
}
```

This query will return the label (visible title of the field in the pipe), type (eg: short text, checkbox etc.), id (a string id of the field) and internal_id (a numeric id of the field) of the start form fields and phase fields .

TIP: You can click the “History” button and save this query as a favorite to get it again faster. You can also do this for other queries/mutations you use frequently.



Execute the query by clicking the execute button () or using the shortcut Ctrl+Enter, and the result will be returned on the right side of the screen.



GraphiQL
Prettify
Merge
Copy
History

```

1 query QueryPipeFields {
2   pipe(id: "302720855") {
3     start_form_fields {
4       label
5       type
6       id
7       internal_id
8     }
9     phases {
10      name
11      id
12      fields {
13        label
14        type
15        id
16        internal_id
17      }
18    }
19  }
20 }
21

```

```

{
  "data": {
    "pipe": {
      "start_form_fields": [
        {
          "label": "This is a startform field",
          "type": "short_text",
          "id": "this_is_a_startform_field",
          "internal_id": "344281507"
        }
      ],
      "phases": [
        {
          "name": "1st Phase",
          "id": "316953706",
          "fields": [
            {
              "label": "This is a 1st phase field",
              "type": "short_text",
              "id": "this_is_a_1st_phase_field",
              "internal_id": "344281582"
            }
          ]
        },
        {
          "name": "2nd Phase",
          "id": "316953707",
          "fields": [
            {
              "label": "This is a 2nd phase field",
              "type": "short_text",
              "id": "this_is_a_2nd_phase_field",
              "internal_id": "344281571"
            }
          ]
        },
        {
          "name": "3rd Phase",
          "id": "316953708",
          "fields": [
            {
              "label": "This is a 3rd phase field",
              "type": "short_text",
              "id": "this_is_a_3rd_phase_field",
              "internal_id": "344281579"
            }
          ]
        }
      ]
    }
  }
}

```

This result contains all the information of the fields necessary to perform queries/mutations that interact with them, most will use the id (string id), but some such as those that handle conditionals use the internal_id (numeric id).

Appendix B: References

No.	Topic	Reference Link
1	Pipefy	Click here
2	Pipefy developers page	Click here
	Pipefy community	Click here
	Pipefy help and support	Click here
1	Overview of Enterprise A2019	Click here
2	Guidance: Building basic A2019 bots	Click here
3	Guidance: Building A2019 action packages	Click here
4	APeople Community Forum	Click here
5	Automation Anywhere University	Click here