

Orchestrator API using http request Activity:

Performs HTTP requests to the Orchestrator API by authenticating under the Robot it is executed on. You can use the GET, POST, PUT, PATCH and DELETE verbs to extract data or manipulate it, and send specific information through JSON. Please note that for each request you might need different rights on the Orchestrator Robot Role, depending on what requests you are doing, and the Robot must be connected to Orchestrator. For example, to run PUT requests on organization units, you need **View**, **Create**, and **Edit** permissions. The activity runs under the Robot which executes it.

Properties

Common

- **ContinueOnError** - Specifies if the automation should continue even when the activity throws an error. This field only supports Boolean values (true, false). The default value is false. As a result, if the field is blank and an error is thrown, the execution of the project stops. If the value is set to true, the execution of the project continues regardless of any error.
- **DisplayName** - The display name of the activity.
- **TimeoutMS** - Specifies the amount of time (in milliseconds) to wait for the activity to run before an error is thrown. The default value is 30000 milliseconds (30 seconds).

Input

- **JSONPayload** - The information you want to send to the indicated Orchestrator endpoint, in a JSON format. The JSON must not be indented. This field supports only strings, string variables or string expressions.
- **Method** - The request method to be used when calling the Orchestrator API. The following HTTP verbs are supported: GET, POST, PUT, PATCH and DELETE. By default, the GET verb is selected.
- **RelativeEndpoint** - The endpoint at which to make requests, relative to your base Orchestrator URL.

Output

- **JSONResponse** - The JSON returned by the HTTP request, as a string variable.
- **StatusCode** - The status code returned by the HTTP request, as an integer variable.

Dependencies:

UiPath.WebAPI.Activities

Get Folder Data from Orchestrator:

Input:

JSONPayload –

Method-GET

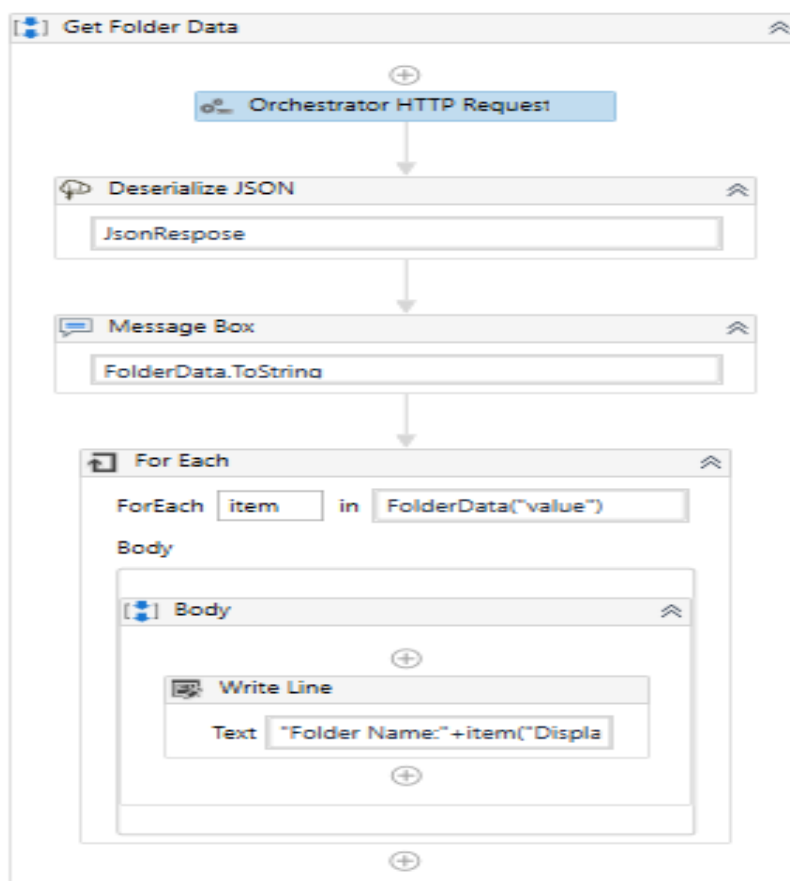
RelativeEndPoint- `"/odata/folders"`

Output:

JSONResponse- JsonResponse

StatusCode- StatusCode

Practical:



Get Process Version from Orchestrator:

Input:

JSONPayload –

Method-GET

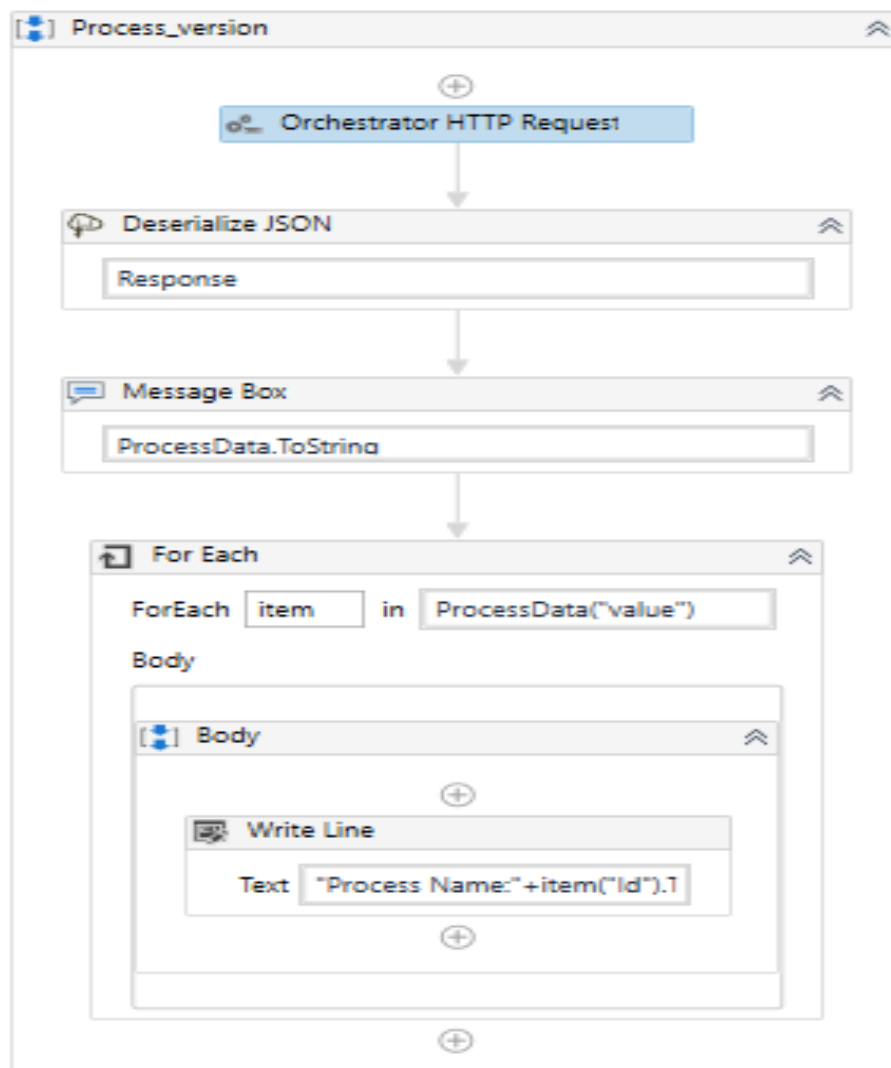
RelativeEndPoint "\odata\Processes"

Output:

JSONResponse- JsonResponse

StatusCode- StatusCode

Practical:



Get RobotLogs:

Input:

JSONPayload –

Method-GET

RelativeEndPoint - `"/odata/RobotLogs"` OR

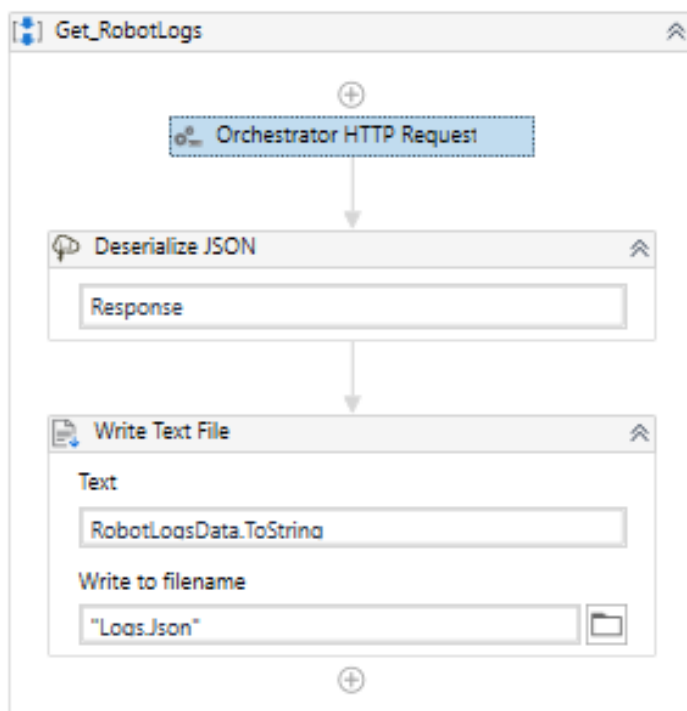
`"/odata/RobotLogs?$filter=RobotName eq 'RobotName'"`

Output:

JSONResponse- JsonResponse

StatusCode- StatusCode

Practical:



Run the Job:

To run the job, we need 2 inputs

1. Release key
2. Robotid

1.To Get Release key:

Input:

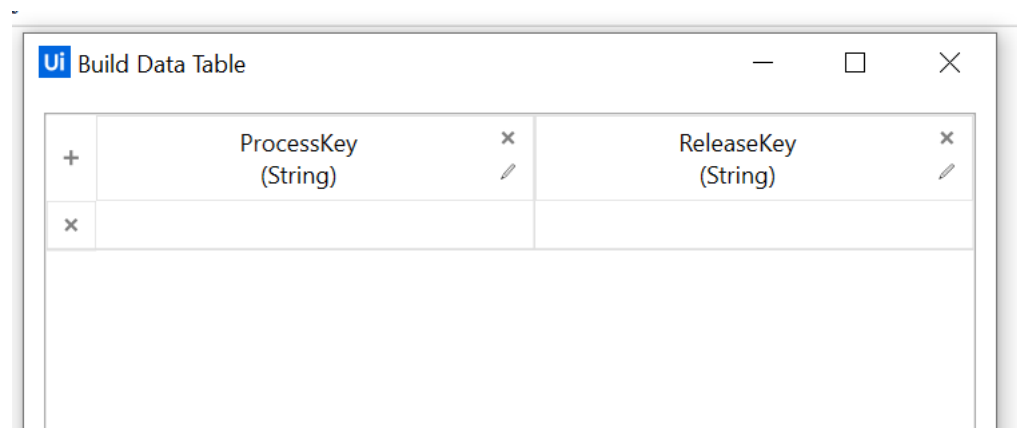
- **JSONPayload** –
- **Method**-GET
- **RelativeEndPoint** – `"/odata/Releases"`

Output:

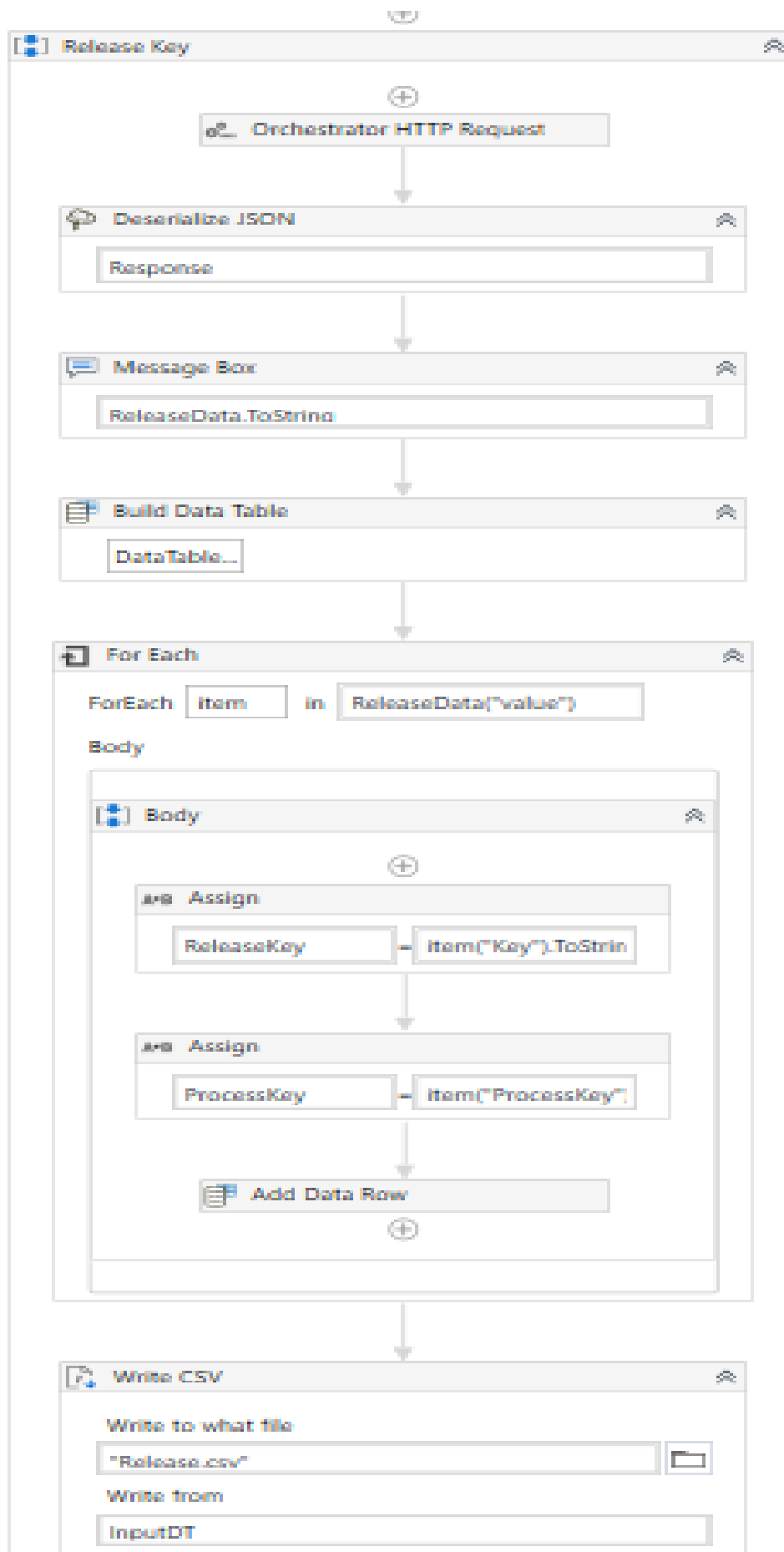
- **JSONResponse**- JsonResponse
- **StatusCode**- StatusCode

Practical:

- 1.Release key and process name, we get from the output.
- 2.Store the 2 values in datatable.
- 3.Write the data into CSV file.



	ProcessKey (String)	ReleaseKey (String)



RobotID

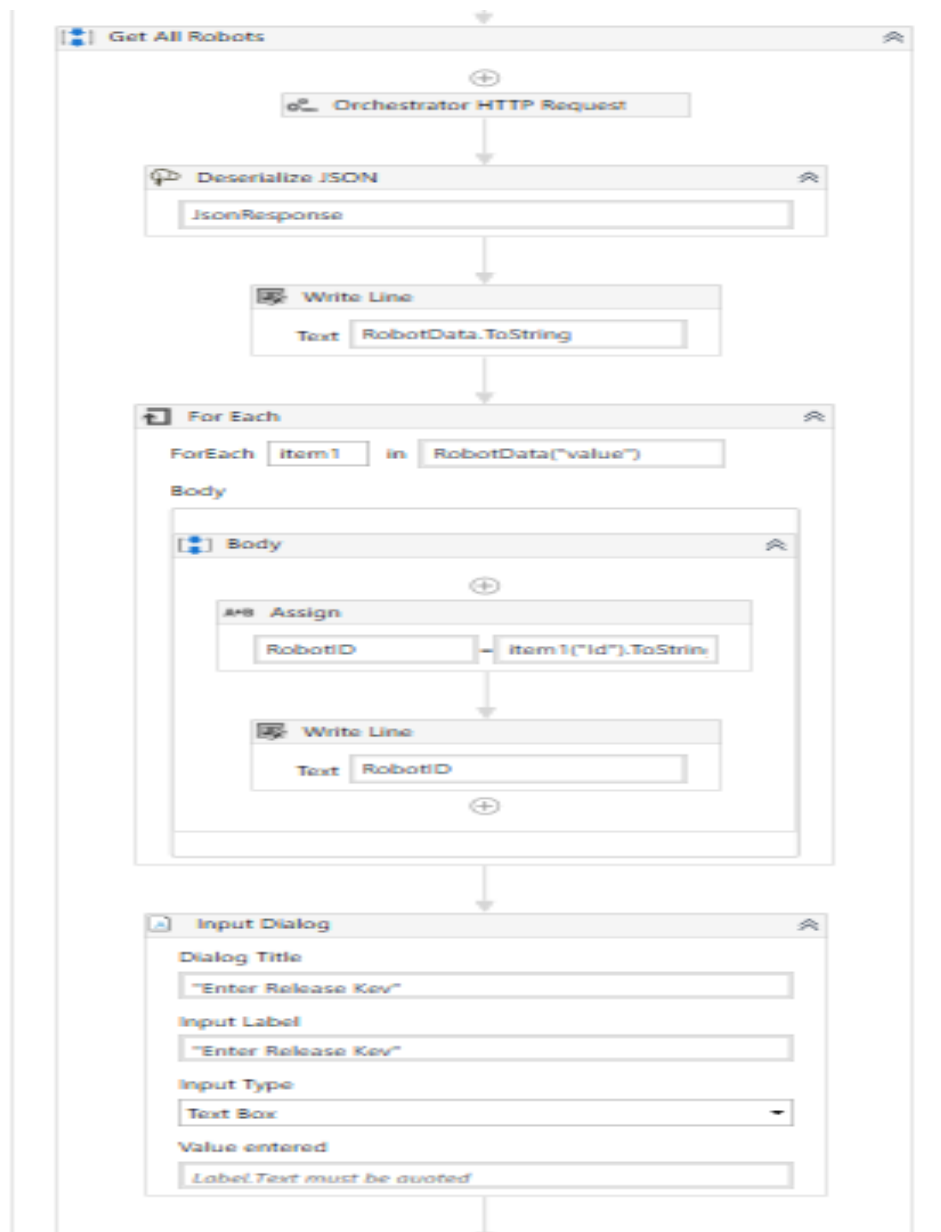
Input:

- **JSONPayload** –
- **Method**-GET
- **RelativeEndPoint** – *"/odata/Robots"*

Output:

- **JSONResponse**- JsonResponse
- **StatusCode**- StatusCode

Practical:



Start Job:

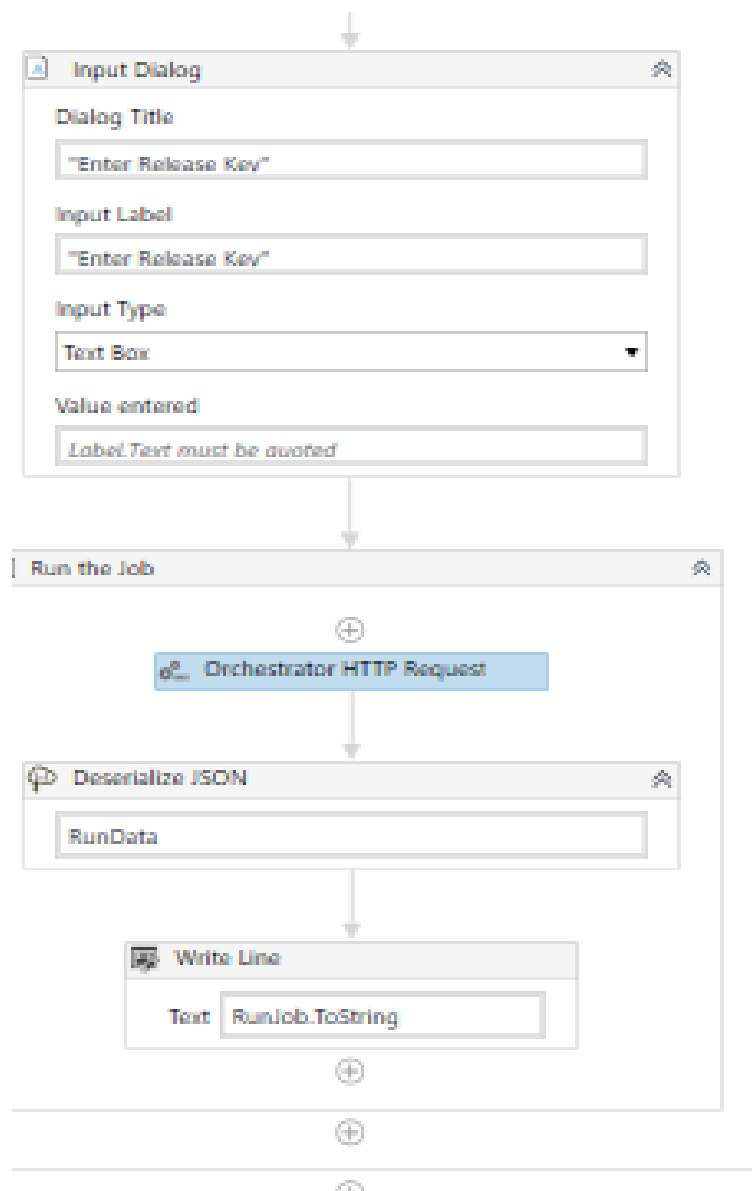
Input:

- **JSONPayload** – "{startInfo:{ReleaseKey: '"+ReleaseKey+"',Strategy: 'Specific',RobotIds:['"+RobotID+"'], JobsCount:0}}"
- **Method**-POST
- **RelativeEndPoint** – "/odata/Jobs/UiPath.Server.Configuration.OData.StartJobs"

Output:

- **JSONResponse**- JsonResponse
- **StatusCode**- StatusCode

Practical:



Create and Get Asset

Create Asset:

Input:

- **JSONPayload** – "{Name: '"+AssetName+"',ValueScope: 'Global',ValueType: 'Text',StringValue: '"+AssetValue+"'}"
- **Method**-**POST**
- **RelativeEndPoint** – "/odata/Assets"

Output:

- **JSONResponse**- JsonResponse
- **StatusCode**- StatusCode

Get Asset:

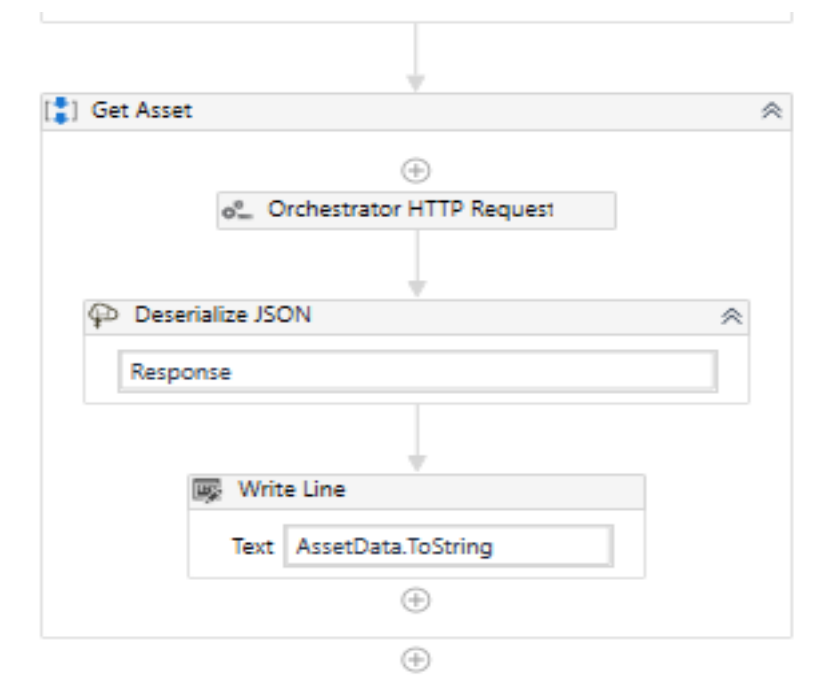
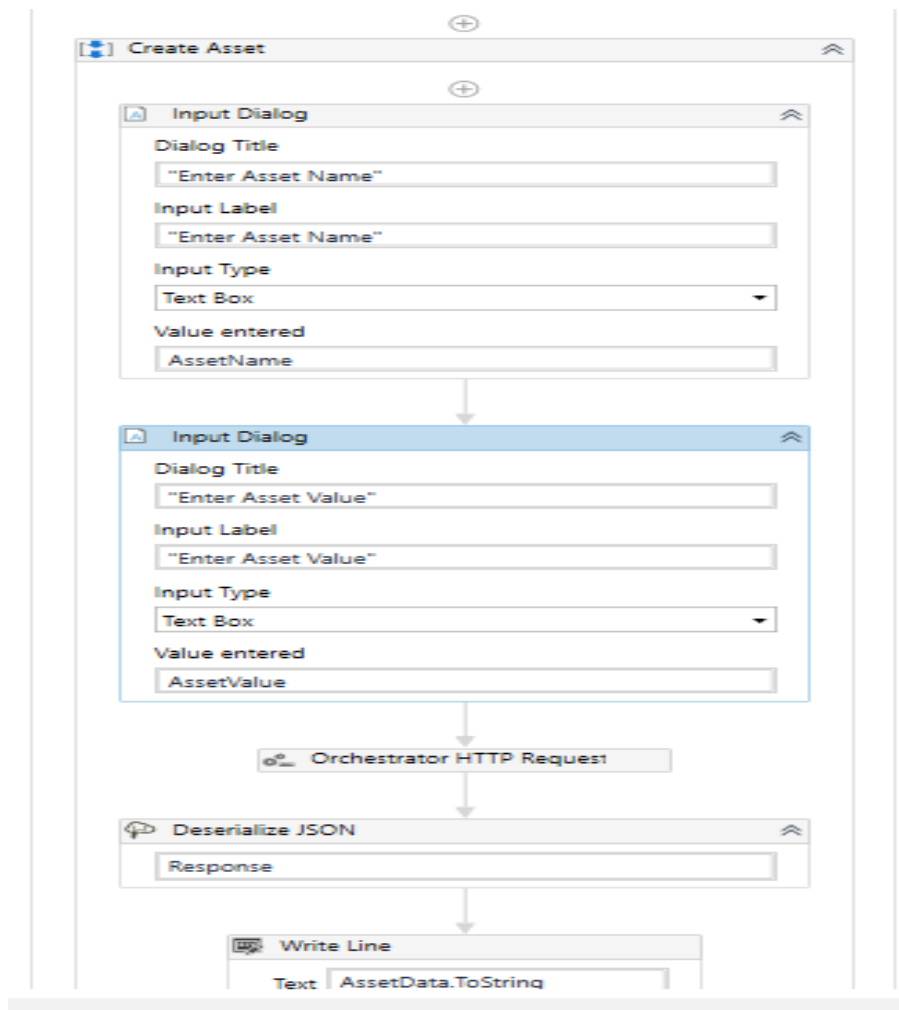
Input:

- **JSONPayload** –
- **Method**-**Get**
- **RelativeEndPoint** – "/odata/Assets"

Output:

- **JSONResponse**- JsonResponse
- **StatusCode**- StatusCode

Practical:



Create and add Queue item:

Case 1:

Input:

- **JSONPayload** – "{itemData:{Name: '"+QueueName+"',Priority:'"+Priority+"',SpecificContent: { '"+Name1+": '"+Value1+"'}}}"
- **Method**-Post
- **RelativeEndPoint** – "/odata/Queues/UiPathODataSvc.AddQueueItem"

Output:

- **JSONResponse**- JsonResponse
- **StatusCode**- StatusCode

Case 2:

Input:

- **JSONPayload** – "{ itemData: { Priority:'"+Priority+"',Name: '"+QueueName+"',SpecificContent: { '"+name1+": '"+value1+"', '"+Name2+": '"+Value2+"'}}}"
- **Method**-Post
- **RelativeEndPoint** "/odata/Queues/UiPathODataSvc.AddQueueItem"

Output:

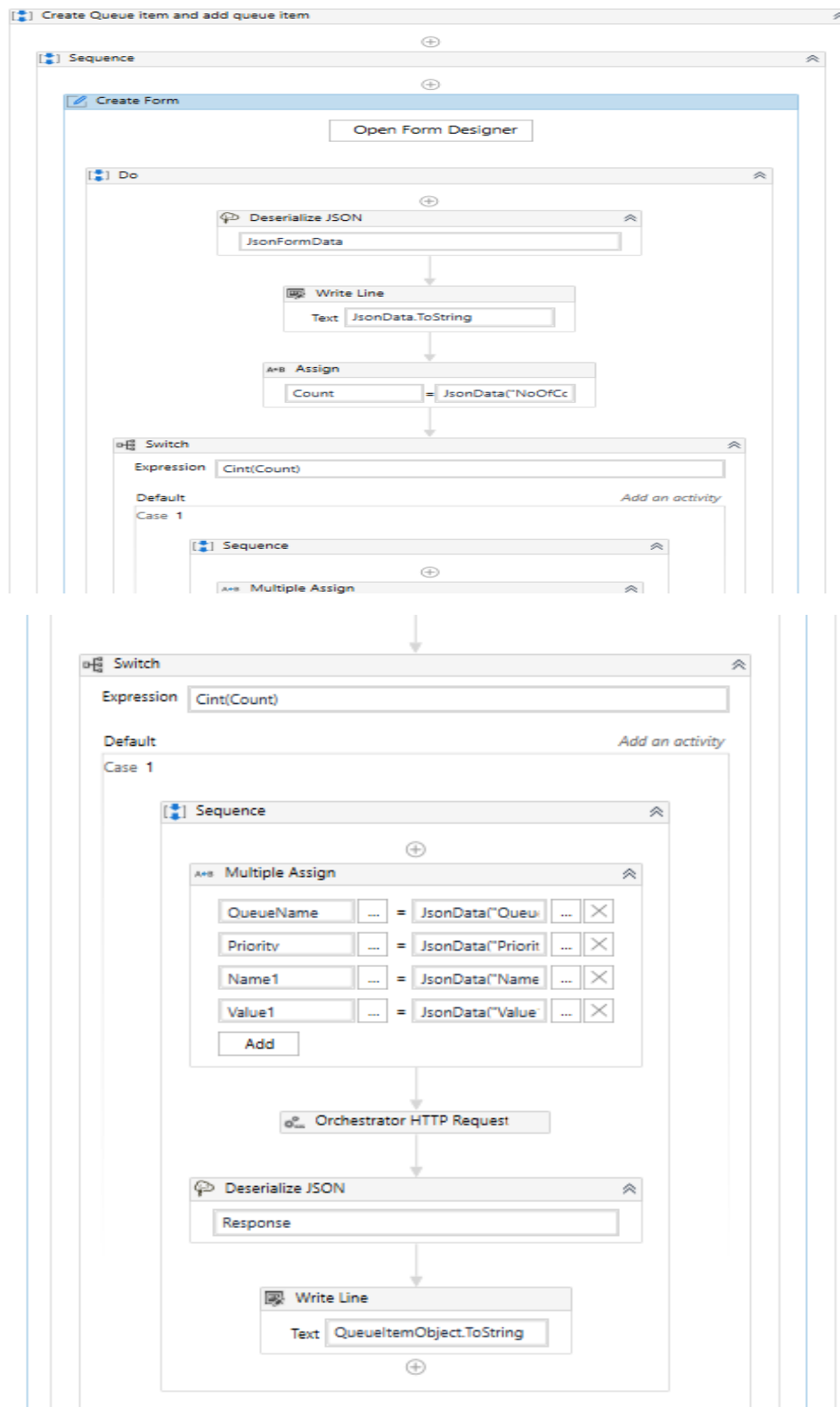
- **JSONResponse**- JsonResponse
- **StatusCode**- StatusCode

Dependencies:

To design form

- **Uipath.Form.Activities**

Practical:



Case 2

