**Orchestrator API’s**

**Solution Design Document**

Version

Date 24 June 2024

**SUMMARY**

|  |  |
| --- | --- |
| Document Owner(s) |  |
| Document Author(s) |  |
| Document Status | Draft |

**SUMMARY**

|  |  |  |
| --- | --- | --- |
| Name | Approval Type | Date Approved |
|  |  |  |

**CHANGE LOG**

|  |  |  |  |
| --- | --- | --- | --- |
| Version # | Implemented By | Document Submission Date | Change Description |
| 1 | Process Architect |  | Original document created and submitted for approval |

**CLIENT SME DETAILS**

|  |  |  |
| --- | --- | --- |
| Name | Role | Email ID |
|  |  |  |

Table of Contents

[**1.** **Objective** 4](#_Toc524367054)

**2.** **Scope for Automation** 4

**3.** **Applications** 4

**4.** **Input Requirements** 4

**5.** **Complexity of the To-Be Process** 6

**6.** **Detailed Solution Design** 6

**7.** **Success Criteria** 22

**8.** **Exception Handling** 23

**9.** **Process Maintenance** 24

**10.** **Automation Operational Arrangements** 24

**11.** **Business Continuity Plan/ Disaster Recovery (DR) Plan** 25

**12.** **Key Assumptions** 25

**13.** **Document Review Guidelines** 25

**14.** **Glossary** 26

**15.** **Appendix** 26

Appendix A: Process Map 26

Appendix B: Process Templates 26

# **Objective**

This document contains the solution design for the **Orchestrator API’s** process. It lays out the high-level “As-Is” process steps as well as the “To-Be” process steps to be automated using the selected Robotic Process Automation (RPA) tool. It also elaborates upon key aspects of the automation that is being implemented for this use case.

* **High Level Overview of As-Is Process:**
* In this process, we create assets, queues, processes, and folders using the Orchestrator HTTP request activity.
* As part of the same process, we retrieve folders, assets, and queues from Orchestrator and store them in an Excel if needed.
* In addition, we create libraries, publish them to Orchestrator, and commit and push them to GitHub.

# **Scope for Automation**

Utilize the Orchestrator HTTP Request activity for communication, ensuring proper authentication and handling of error scenarios. Develop reusable libraries in UiPath Studio and publish them to Orchestrator. Implement version control by committing and pushing UiPath projects (including libraries) to a GitHub repository, automating this process as part of your development workflow.

# **Applications**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Application Name& Version** (including any upgrade information) | **External / Internal** | **Type** | **Credentials required for BOT ID** | **Read/ Write** |
| Orchestrator | Internal | Cloud Based Application | Yes | Read & Write |
| Server Management Studio | Internal | Desktop Application | Yes | Read & Write |
| GitHub | Internal | Web Application | Yes | Read & Write |
| Excel | Internal | Desktop Application | No | Read & Write |

# **Input Requirements**

The inputs and pre-requisites detailed out below, need to be present for the Bot to function in the desired manner.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Type** | **Format** | **Name** | **Description** | **Location** | **Initial Value** |
| 1 | Folder iD’s ,Asset Keys,Names | Xlsx | Storing the required details (data) of the processes | The Excel file will contain Folder IDs, names, asset names, asset keys, queue names, etc., which we can use as input. | Desktop | Folder Id and Key |

**4.1 Packages Required**

|  |  |
| --- | --- |
| **Package Name** | **Description** |
| UIpath.WebAPI | It is used for API’s. From this we get the HTTP request |
| UiPath. Database | It is used to connect the database and some other related to the SQL |

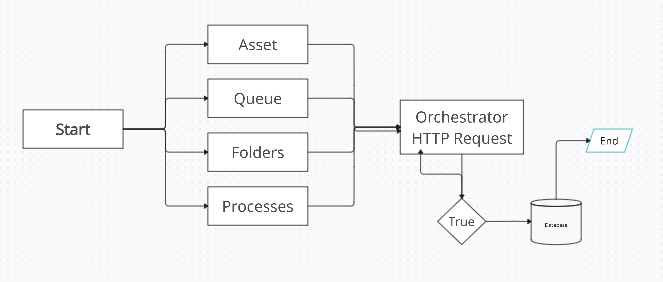
# **Complexity of the To-Be Process**

This process has been categorized as a ‘**Simple**’ process from an RPA Implementation perspective.

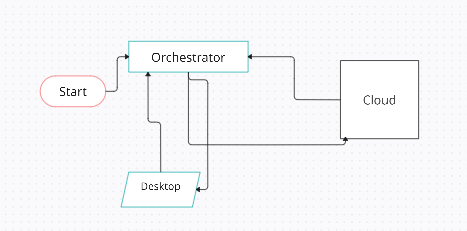
# **Detailed Solution Design**

**High-Level Design**

The following sections capture the business logic necessary to navigate through the process.



**Network Design**



It provides a graphical representation of the components and structure of a network, such as routers and switches.

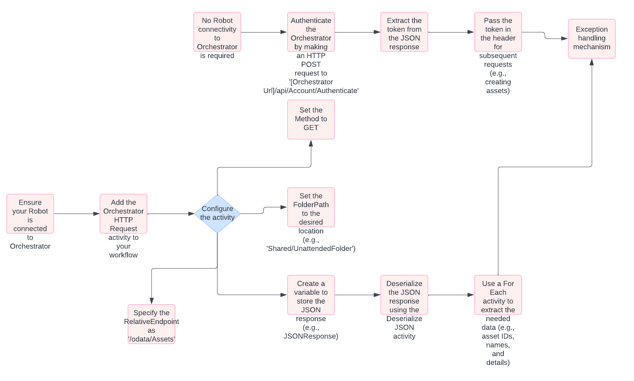
**6.1 Creating and Retrieving the Assets**

|  |  |  |
| --- | --- | --- |
| **Steps** | **Description** | **Activity Name** |
| A1 | Create assets using the HTTP POST method with data types such as integer, text, Boolean, and credentials in an Orchestrator HTTP request. Store the results in a string variable. | Orchestrator HTTP request |
| A2 | To retrieve assets using the HTTP GET method and store them in a string variable, we will use the filter function | Orchestrator Http request |
| A3 | After receiving the response from the Orchestrator HTTP request, it will convert the string to a JObject and provide structured data. | Deserialize JSON |
| A4 | After Retrieving each data from the response, it will be stored into the database | Insert Database |

**6.2 Creating and Retrieving the Queues**

|  |  |  |
| --- | --- | --- |
| **Steps** | **Description** | **Activity Name** |
| A1 | To create queues using the HTTP POST method and add items to each queue in an Orchestrator HTTP request, store the results in a string variable. | Orchestrator HTTP request |
| A2 | To retrieve queues using the HTTP GET method and obtain the items related to each queue in an Orchestrator HTTP request, store the results in a string variable. | Orchestrator Http request |
| A3 | After receiving the response from the Orchestrator HTTP request, it will convert the string to a JObject and provide structured data. | Deserialize JSON |

**7.Final Daigram**



**8. Database Schema**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Database Name | Table Name | Column Names | Size | Constraints |
| Master | GetAsset1 | AssetName  AssetKey  AssetId | Varchar(20)  Varchar(50)  Varchar() | Yes |
| Master | Folders | DisplayName  Id | Varchar(20)  Varchar(20) | yes |

# **9. Exception Handling**

Exceptions handled by robotic process automation can primarily be classified into system and business exceptions. The automation under discussion handles each error type in a standardized manner.

In general, if the Bot encounters any unexpected scenario, the Bot will generate an exception message and inform the same to account team. The Bot will generate exception email, with mitigating actions specified below.

**Business Exceptions**

All known business exceptions are accounted for within the solution design. If an unexpected scenario is found, the Bot will log an exception in UiPath for that record and inform the same to account Team.

Any unexpected business scenarios that the Bot encounters will cause the Bot to skip the case, leaving it open and informing the Account team.

For any business exception, then:

1. **Action taken by automation**: Error captured in the Exception Report for the business exceptions
2. **Reporting Type**: The Exception message for the business scenario will be preserved within the Exception Report.
3. **Manual Action required**:
4. Account Team Supervisor: Manual review recommended if business exception occurs frequently
5. Bot Controller: None

**Known Business Exceptions:**

|  |  |  |
| --- | --- | --- |
| **Exception Scenario** | **Exception Status Message** | **Overcome Exception** |
| Data is not found in Database | The excepted data is not found | To overcome this once check the data and add if necessary |

**System Exceptions**

System Exception such as Retail application not working is considered in the SDD. If an unexpected scenario is found, the Bot will log an exception in UiPath for that record and inform the same to Trading Team.

|  |  |  |
| --- | --- | --- |
| **Exception Scenario** | **Exception Status Message** | **Overcome Exception** |
| The Endpoint is not accessible | The Endpoint is not accessible, and it thrown error code as 400 | Correct the URL which we give as endpoint |
| Database not accessible | Connection between UiPath and database not accessible | Connect the database |

# **9. Process Maintenance**

We should connect the UiPath with the Orchestrator and Database. Have to give the correct URL’s which we need to get or post the data into the Orchestrator

# **10. Automation Operational Arrangements**

|  |  |
| --- | --- |
| What part will the Business play within the end-to-end process? | Review |
| Will the Business re-submit exception cases to the solution? | Possibly |

# **11. Business Continuity Plan/ Disaster Recovery (DR) Plan**

In case of complete failure of the Bot, the team will need to revert to manual process.

* Operations team will perform the “Process” process manually, until notified
* RPA Capability will be responsible for operational tracking, monitoring, maintenance. In case of any temporary outages or planned downtime, the COE will communicate to business and expedite restoration of services.
* Process owners are expected to systematically review and audit the results to ensure the functional value is provided.
* Changes to host systems to be proactively communicated to members of the RPA Capability.
* The process owner will receive notification emails for all failures.

# **12.Document Review Guidelines**

This document is to be reviewed every 12 months and updated with relevant process and system changes to reflect the current process and automation functionality. This review should be conducted by a nominee of the document owner and signed off by the document owner.

**13.Glossary**

|  |  |
| --- | --- |
| **Term** | **Description** |
| RPA | Robotics Process Automation |
| UiPath | UiPath |

# **Appendix**

## Appendix A: Process Map

## Appendix B: Process Templates

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.** | **File Name** | **Type** | **Purpose** | **Attachment** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |