Scientific Research and Reports
2024 Volume 1, Issue 1
DOI: 10.59657/2996-8550.brs.24.001

BRS BioRes Scientia
Value your Research

Research Article

Open Access

Seamless Integration of Automation Anywhere (RPA) Bots into Organizational Systems and Workflows using the A360 Bot Deploy API

Sai Madhur Potturu*

Robotics Center of Excellence (CoE), Zoetis Inc., Parsippany, United States.

Abstract

This paper presents a solution for the seamless integration of Automation Anywhere (RPA) bots into various workflows and processes within an organization. The integration is achieved by leveraging the A360 Bot deploy API, allowing workflows to trigger bots and utilize their automation capabilities effectively. This integration allows for the seamless transfer of work items between distinct phases of the workflow, streamlining and automating organizational processes.

By seamlessly integrating Automation Anywhere with established applications, organizations can take their RPA adoption to the next level. This paper investigates the triggering process of RPA bots by utilizing the A360 Bot deploy API, with and without passing inputs to the bot. It also explores the advantages of leveraging the A360 Bot deploy API to achieve seamless integration between workflow and RPA applications.

By connecting Automation Anywhere bots with various systems, organizations can automate various steps within their workflows, eliminating the need for manual intervention. This integration empowers users to enhance productivity and efficiency by automating tasks within their existing applications.

Keywords: robotic process automation (RPA); automation anywhere, automation 360, bot deploy API, REST API; bot deployment; process optimization; enhance efficiency

Introduction

Automation Anywhere is a leading player in the field of Robotic Process Automation (RPA) and Intelligent Automation [1,2,3]. Their cloud-based platform, A360, offers a reliable and user-friendly solution for building automation solutions. It enables seamless integration with various tools and technologies, supporting process improvement, process transformation, and digital transformation within organizations. Automation Anywhere plays a pivotal role in enabling the integration of systems and automation within organizations [3]. This objective can be accomplished through two distinct approaches: integrating multiple systems into Robotic Process Automation (RPA) and integrating RPA into different systems. When integrating multiple systems in RPA, an RPA solution is developed to seamlessly interact with various systems, applications, or databases. Acting as a central orchestrator, the RPA tool enables end-to-end process automation involving multiple systems. This approach empowers organizations to streamline their operations and enhance efficiency [4].

On the other hand, the integration of RPA in different systems entails embedding RPA capabilities within individual systems or applications [5]. In this scenario, each system or application incorporates RPA functionality within its own ecosystem, eliminating the need for a centralized RPA orchestrator [6]. This integration can be achieved by incorporating the A360 Bot deploy API into system workflows, allowing workflows to trigger RPA Bots, and enabling automation capabilities within the workflows. This study explores the process of triggering RPA bots by utilizing the A360 Bot deploy API, both with and without passing inputs to the bots. The A360 Bot Deploy API refers to a REST API (Application Programming Interface) provided by Automation Anywhere for their A360 platform [7]. This API allows users to deploy bots, which are automated tasks or processes, within the A360 environment. The A360 Bot Deploy API simplifies the deployment and management of bots within the A360 platform, providing efficiency, consistency, workflow integration, scalability, flexibility, and centralized control for organizations leveraging automation in their processes.

© 2024 Sai Madhur Potturu.

The decentralized approach, "integration of RPA in different systems" presents opportunities to enhance system-specific automation and drive comprehensive process optimization. It enables seamless integration of A360 bots into workflows, allowing organizations to enhance their RPA adoption and improve efficiency in their operations [8].

Solution

The solution explains the process of triggering an RPA Bot using the Bot Deploy API, both with and without passing inputs [7]. The first part focuses on the essential prerequisites and user permissions needed to create RPA bots and access the Bot Deploy API. In the second part, the triggering process is explained by creating two distinct bots - a Dispatcher bot and a Performer bot. The Dispatcher bot triggers the Performer bot, utilizing the A360 Bot Deployment API. On the other hand, the Performer bot executes the automation workflow when triggered by the Dispatcher bot. The Dispatcher bot is developed using a task bot in A360 solely for demonstrating the triggering process of the Performer bot using the Bot Deploy API. In real-world scenarios, the A360 Bot Deploy API is directly integrated into the enterprise system workflows to trigger the bots. The performer bot is a main task that can execute various subtasks within the automation process workflow. The main task and subtasks both are developed using the task bot in A360 [9], there is no difference in the interface or availability of commands/packages in the main task or subtask. It is just a classification depending on their usage.

User Permissions

Before developing and deploying the bots, ensure the following user permissions must be configured.

Performer Bot & Dispatcher Bot

To create a Performer and Dispatcher bot in A360, the user must be registered in the A360 control room and assigned a bot creator license and the "AAE_Bot Developer role".

Authentication API

Authentication API is used to authenticate a registered user in the A360 control room [10]. All API calls must contain an authentication token from Authentication API in the header section [8]. The Authentication API generates, refreshes, and manages JSON web tokens that are required for authorization in all A360 enterprise control room APIs. Users who have Single Sign-On (SSO) access to A360 Control Room can utilize an API key for authentication. On the other hand, users without SSO access can use their A360 Control Room password for authentication. To grant users permission to generate an API key, create a custom role and include the 'Generate API-key permissions' within that role. Once created, assign this custom role to the respective user.

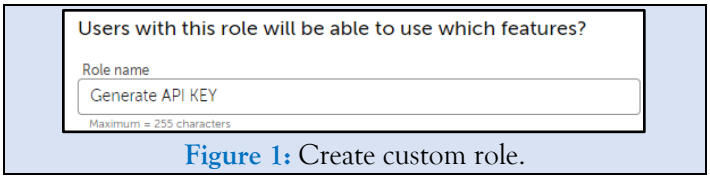

Figure 1: Create custom role.

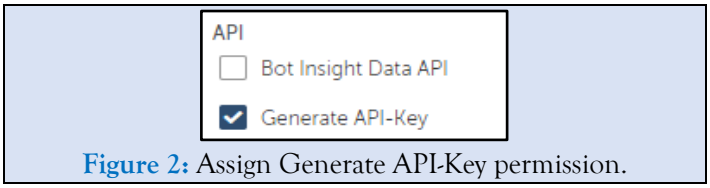

Figure 2: Assign Generate API-Key permission.

Bot Deploy API

The Bot Deploy API deploys bots on an unattended bot runner VDI [7]. Before deploying the bot, the user needs to be authorized through the Authentication API, and the authentication token must be included in the request header.

For triggering the Performer bot on an unattended bot runner machine, the user requires the following permissions.

- The performer bot must be in a public workspace.
- The user must have the "View and Run my bots" feature permission.
- The user must have "Run and schedule" permissions for the folders containing the bots.
- The user must have access to Bot Runner licensed users.
- The user must have access to either a default device or a device pool.

Triggering Performer Bot with No Input Parameters

The process of triggering the Performer bot by the Dispatcher bot without passing inputs is explained in detail.

Performer Bot

The Performer bot can receive inputs from the caller task and return outputs to the caller task. In this scenario, the Dispatcher bot triggers the Performer bot without passing any inputs, thereby eliminating

© 2024 Sai Madhur Potturu.

the need to configure input type variables in the Performer bot.

Dispatcher Bot

The Dispatcher bot initiates the process by first calling the Authentication API to authenticate the Control Room user. After successful authentication, it proceeds to call the Bot Deploy API to deploy the Performer bot.

Authentication API

The initial step of the Dispatcher bot involves calling the Authentication API to authorize the user through the "Rest Web Services-POST method" action.

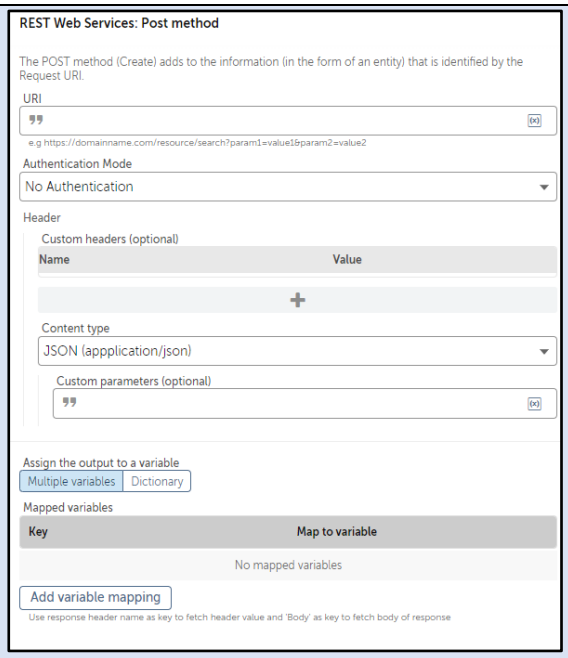

Figure 3: A360 Rest Web Services Post Method Action

To trigger the Authentication API, the Dispatcher bot requires the following endpoints and configurations.
URL: The URL is a combination of the Control Room URL and "/v1/authentication".

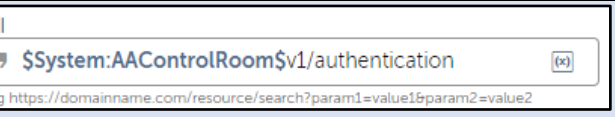

Figure 4: Authentication API URL

Authentication Mode: Choose the "No Authentication" option from the Authentication model dropdown.

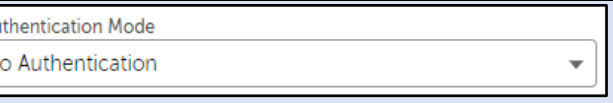

Figure 5: Authentication API Authentication Mode

Headers:

Content Type: Set the content type for the Authentication API as JSON.

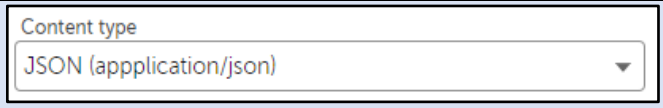

Figure 6: Authentication API Content type

Custom Parameters: User credentials are passed as a request body schema in JSON format.

```
{
 "username": "jdoe", // Replace with the username (e.g., domain/userid or email address)
 "password": "mypassword@123", // Replace with the user's password.
 "apiKey":  "Vie;Z:IvtAhY0NU7baRLYEeIYUJSKO", // Use the API key as an alternative to the password (optional)
 "multiLogin": true // Set to true to allow multiple user logins or false to disallow [11].
}
```

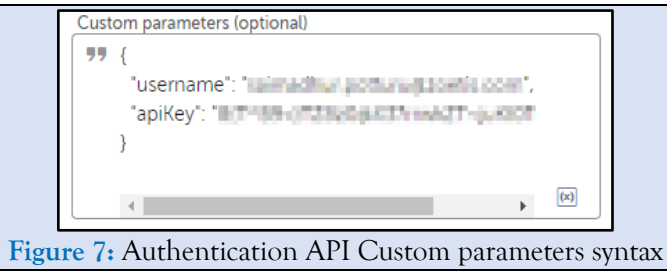

Figure 7: Authentication API Custom parameters syntax

Output: The Authentication API Provides four potential responses.
200: Success
This response contains a JSON object with the JWT (token) generated for the control room user. It also includes details about the user's roles, permissions, and licenses.
400: Bad Request
This response contains a JSON object with an error code, error message, and additional details about the error.
401: Authentication required
This response contains a JSON object with an error code, error message, and additional details about the error.
500: Internal server error
This response contains a JSON object with an error code, error message, and additional details about the error.
Extract Authentication Token

© 2024 Sai Madhur Potturu.

In the Dispatcher bot, the Authentication API's output response is stored in a Dictionary variable. The token is then extracted from this dictionary using string manipulation operations.

Figure 8: A360 String Extract text action

Bot Deploy API

Once the token is retrieved, the Dispatcher bot proceeds to trigger the Performer bot by calling the Bot Deploy API using the "Rest Web Services-POST method" action. To trigger the “Bot Deploy API,” the Dispatcher bot requires the following endpoints and configurations:

URL: The URL is a combination of the Control Room URL and "/v3/automations/deploy".

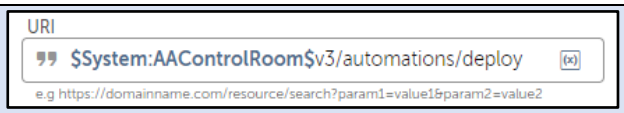

Figure 9: Bot Deploy API URL

Authentication Mode: Choose the "No Authentication" option from the Authentication model dropdown.

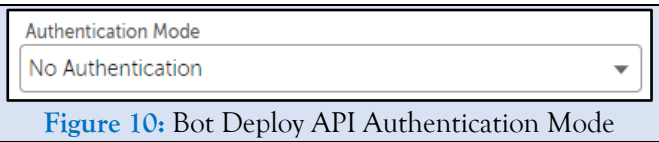

Figure 10: Bot Deploy API Authentication Mode

Headers:

X-Authorization: Pass the Token variable retrieved from the Authentication API response.

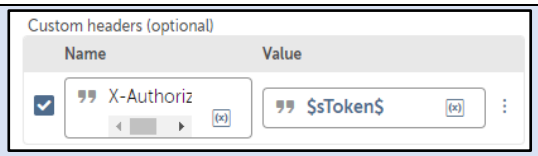

Figure 11: Bot Deploy API Content type

Content Type: Set the content type for the Authentication API as JSON

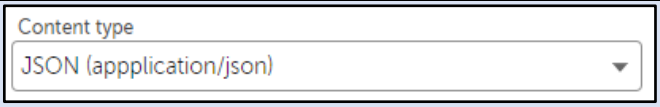

Figure 12: Bot Deploy API Content type

Custom Parameters: The Performer bot file ID and the Control Room bot user ID with an unattended bot runner license are passed as a request body schema in JSON format.

```
{
 "fileId": 83, // Performer bot ID in A360 Control Room. This ID can be found in the URL when clicking on the processor bot to view or edit.
 "runAsUserIds": [
   3 // The unattended bot user ID can be found in the URL when clicking on the user to view or edit.
 ]
}
```

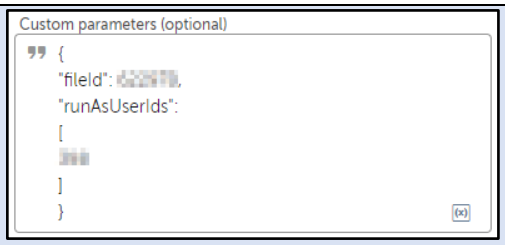

Figure 13: Bot Deploy API Custom parameters syntax

Output: The Bot Deploy API provides five potential responses.

- 200: Success
- Returns a Deployment ID that can be used to check the deployment's status using the callback URL.
- 400: Bad Request
- This response contains a JSON object with an error code and error message.
- 401: Authentication required
- This response contains a JSON object with an error code and error message.
- 404: Not Found
- This response contains a JSON object with the error message.
- 500: Server error
- This response contains a JSON object with an error code and error message.

© 2024 Sai Madhur Potturu.

Triggering Performer Bot with Input Parameters

The process of the Dispatcher bot triggering the Performer bot and passing several types of inputs is explained in detail.

Performer Bot

In this scenario, the Dispatcher bot triggers the Performer bot and passes values for predefined variables in the Performer bot. To achieve this, the input type variables must have been previously created in the Performer bot, enabling the Dispatcher bot to pass values when it triggers the Performer bot using the Bot Deploy API.

Dispatcher Bot

The Dispatcher bot initiates the process by first calling the Authentication API to authenticate the Control Room user. After successful authentication, it proceeds to call the Bot Deploy API to deploy the Performer bot.

Authentication API

In this scenario, the initial step of the Dispatcher bot involves calling the Authentication API to authorize the user through the "Rest Web Services-POST method" action, remains the same as the first step of the Dispatcher bot in the previous scenario of " Triggering Performer Bot with No Input Parameters ".

Extract Authentication Token

In this scenario, the initial step of the Dispatcher bot involves calling the Authentication API to authorize the user through the "Rest Web Services-POST method" action, remains the same as the first step of the Dispatcher bot in the previous scenario of " Triggering Performer Bot with No Input Parameters ".

Bot Deploy API

Once the token is retrieved, the Dispatcher bot proceeds to trigger the Performer bot by calling the Bot Deploy API using the "Rest Web Services-POST method" action.

To trigger the “Bot Deploy API,” the Dispatcher bot requires the following endpoints and configurations:

URL: The URL is a combination of the Control Room URL and "/v3/automations/deploy".

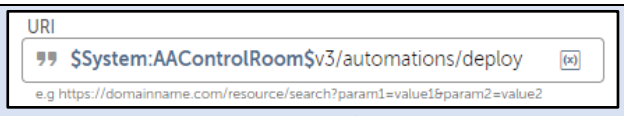

Figure 14: Bot Deploy API URL

Authentication Mode: Choose the "No Authentication" option from the Authentication model dropdown.

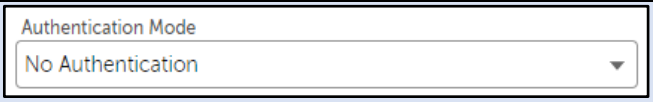

Figure 15: Bot Deploy API Authentication Mode

Headers

X-Authorization: Pass the Token variable retrieved from the Authentication API response.

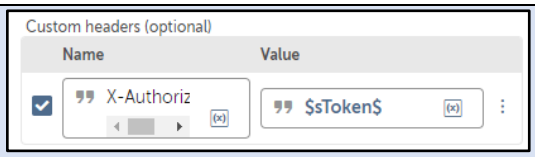

Figure 16: Bot Deploy API Content type

Content Type: Set the content type for the Authentication API as JSON

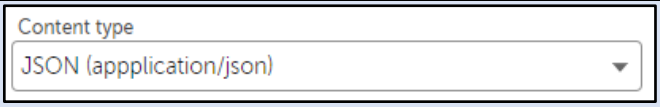

Figure 17: Bot Deploy API Content type

Custom Parameters

The 'fileId' parameter is an Integer type. It should be provided as part of the request body schema in JSON format to represent the File ID of the performer bot.

"fileId": 83, // Performer bot ID in A360 Control Room.

Include the 'runAsUserIds' parameter as an array of integers in the JSON request body to specify the bot IDs for deploying the performer bot.

```
"runAsUserIds": [
  3 // Unattended bot user ID
]
```

The "botInput" parameter allows passing input values to the Performer bot. It supports various types of inputs, such as "STRING", "NUMBER", "BOOLEAN", "FILE", "ITERATOR", "LIST", "DICTIONARY", "TABLE", "VARIABLE", "CONDITIONAL", "WINDOW", "TASKBOT", "DATETIME", "UIOBJECT", "RECORD", "EXCEPTION", "CREDENTIAL", "COORDINATE", "IMAGE", "REGION", "PROPERTIES", "TRIGGER", "CONDITIONALGROUP", "FORM",

© 2024 Sai Madhur Potturu.

"FORMELEMENT", "HOTKEY", and "WORKITEM." The default attribute is set to "STRING" [7][12].

Among these types, "STRING", "NUMBER", "BOOLEAN", "LIST" and "DICTIONARY" are commonly used in automation processes. The process of creating variables in the Performer bot and passing values to these variables using the Bot deploy API in the Dispatcher bot is demonstrated for each variable type.

"STRING":

Create a string variable in the Performer bot and use its name in the Bot deploy API to pass values from the Dispatcher bot.

Performer Bot

Two string type variables "vStrString1" and "vStrString2" are created in the performer bot.

Figure 18: Create vStrString1 variable

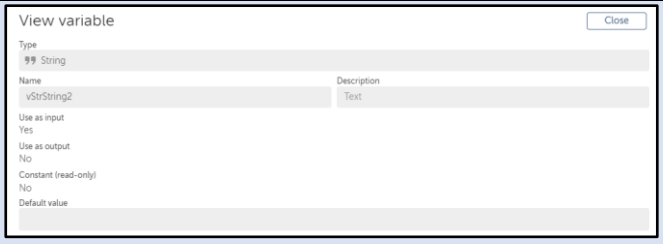

Figure 19: Create vStrString2 variable

Dispatcher Bot

In the Bot deploy API's "botInput" parameter, use JSON format to specify the variable types and values for "vStrString1" and "vStrString2" variables.

Syntax

```
"botInput": {// parameter
      "vStrString1": {    // variable name
            "type": "STRING", // variable type
             "string": "Hello"    // variable value
              },
      "vStrString2": {// variable name
            "type": "STRING", // variable type
             "string": "World!"   // variable value
              },
```

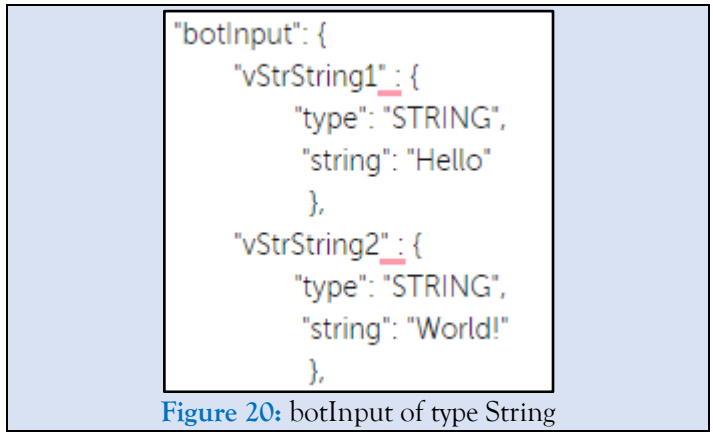

Figure 20: botInput of type String

"NUMBER":

Create a Number variable in the Performer bot and use its name in the Bot deploy API to pass values from the Dispatcher bot.

- **Performer Bot:**

A number type variable "vNumRandomID" is created in the performer bot.

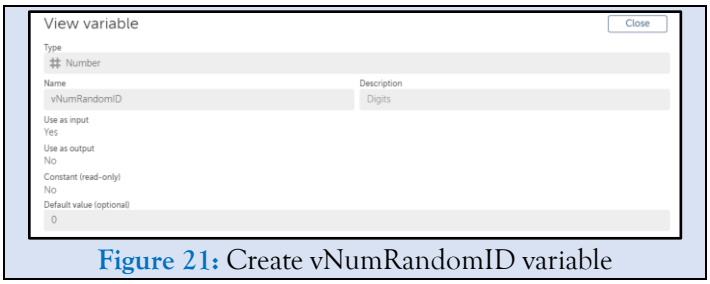

Figure 21: Create vNumRandomID variable

- **Dispatcher Bot:**

In the Bot deploy API's "botInput" parameter, use JSON format to specify the variable type and value for the "vNumRandomID" variable.

- **Syntax**

```
"botInput": {// Parameter
      "vNumRandomID": {// variable name
            "type": "NUMBER", // variable type
             "number": 500     // variable value
              },
```

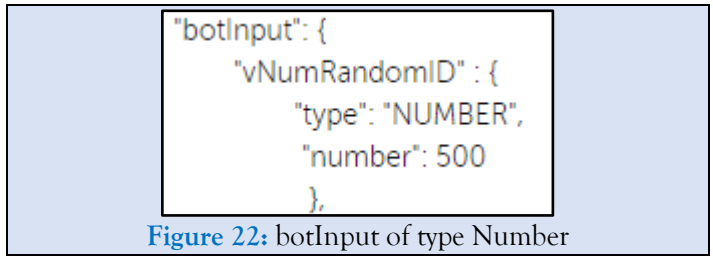

Figure 22: botInput of type Number

"BOOLEAN":

Create a Boolean variable in the Performer bot and use its name in the Bot deploy API to pass values from the Dispatcher bot.

© 2024 Sai Madhur Potturu.

- **Performer Bot:**

A Boolean type of variable “vBlnFlag” is created in the performer bot.

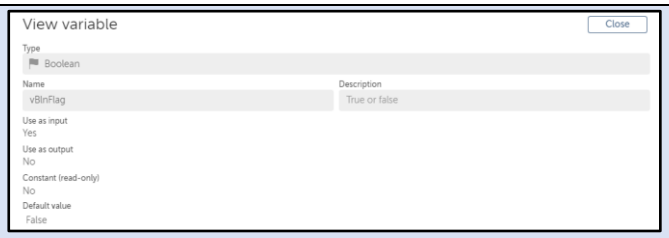

Figure 23: Create vBlnFlag variable

- **Dispatcher Bot:**

In the Bot deploy API's "botInput" parameter, use JSON format to specify the variable type and value for the "vBlnFlag" variable.

- **Syntax**

```
"botInput": {// Parameter
      "vBlnFlag": {// variable name
            "type": "BOOLEAN", // variable type
             "boolean": True     // variable value
              },
```

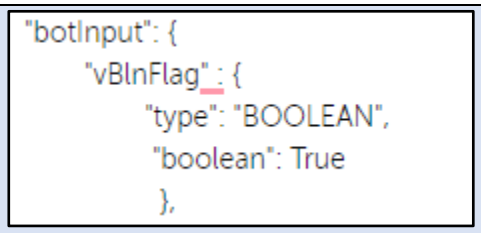

Figure 24: botInput of type Boolean

“LIST”:

Create a List variable in the Performer bot and use its name in the Bot deploy API to pass values from the Dispatcher bot.

Performer Bot:

A List type variable “vListNames” is created in the performer bot.

Figure 25: Create vListNames variable.

- **Dispatcher Bot:**

In the Bot deploy API's "botInput" parameter, use JSON format to specify the variable type and values for the "vListNames" variable.

- **Syntax**

```
"botInput": {// parameter
      "vListNames": {// variable Name
                  "type": "LIST", //variable type
                  "list": [ // variable values
                        {
            "type": "STRING", // value 1 type
            "string": "Apple"   // value 1
             },
                  {
            "type": "STRING", // value 2 type
            "string": "Banana" // value 2
             },
                  {
            "type": "STRING", // value 3 type
            "string": "Grapes" // value 3
             }
                  ]
                  },
```

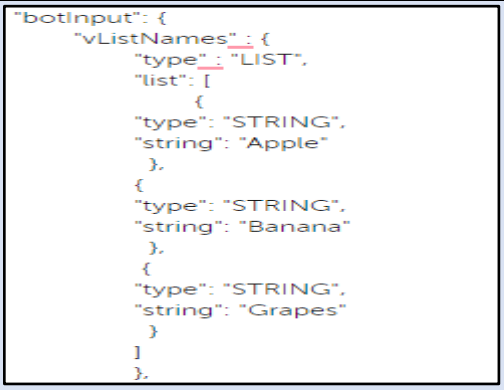

Figure 26: botInput of type List

“DICTIONARY”:

Create a Dictionary variable in the Performer bot and use its name in the Bot deploy API to pass values from the Dispatcher bot.

- **Performer Bot:**

A Dictionary type variable “vDicUserDetails” is created and its “Keys” are defined in the performer bot.

© 2024 Sai Madhur Potturu.

Figure 27: Create vDicUserDetails variable

Figure 28: botInput of type Dictionary

- **Dispatcher Bot:**

In the Bot deploy API's "botInput" parameter, use JSON format to specify the variable type and values for the "vDicUserDetails" variable.

Syntax

```
"botInput": {// parameter
      "vDicUserDetails": {// variable name
            "type": "DICTIONARY", // variable type
            "dictionary": [ // variable value

        {
        "key": "Name", // key 1
        "value": {
                "type": "STRING", // value 1 type
                "string": "Specify value" // value 1
                }
},
        {
        "key": "Email", // key 2
        "value": {
                "type": "STRING", // value 2 type
                "string": "Specify value"   // value 2
                }
},
        {
        "key": "Department", // key 3
        "value": {
                "type": "STRING", // value 3 type
                "string": "Specify value" // value 3
                }
}
                  ]
            }
}
```

The syntax for custom parameters, which includes "fileId," "runAsUserIds," and "botInput" with several types of variables, is as follows:

```
{
"fileId": 622978,
"runAsUserIds": [
398
],
"botInput": {
      "vStrString1": {
            "type": "STRING",
            "string": "Hello"
             },
      "vStrString2": {// variable name
            "type": "STRING",
            "string": "World!"
             },
      "vNumRandomID": {
            "type": "NUMBER",
            "number": 500
             },
      "vBlnFlag": {
            "type": "BOOLEAN",
            "boolean": True
             },
      "vListNames": {
                  "type": "LIST",
                  "list": [
                        {
            "type": "STRING",
            "string": "Apple"
             },
                  {
            "type": "STRING",
            "string": "Banana"
             },
                  {
            "type": "STRING",
            "string": "Grapes"
```

© 2024 Sai Madhur Potturu.

```
            }
                ]
                },
    "vDicUserDetails": {
        "type": "DICTIONARY",
        "dictionary": [
                        {
                          "key": "Name",
                          "value": {
                              "type": "STRING",
                              "string": "username"
                                }
                         },
                        {
                          "key": "Email",
                          "value": {
                           "type": "STRING",
                           "string": "user email id"
                                    }
                         },
                        {
                          "key": "Department",
                          "value": {
                          "type": "STRING",
                          "string": "user department"
                        }
                      }
                    ]
            }
        }
}
```

Output: The Bot Deploy API provides five potential responses.

- 200: Success
- Returns a Deployment ID that can be used to check the deployment's status using the callback URL.
- 400: Bad Request
- This response contains a JSON object with an error code and error message.
- 401: Authentication required
- This response contains a JSON object with an error code and error message.
- 404: Not Found
- This response contains a JSON object with the error message.
- 500: Server error
- This response contains a JSON object with an error code and error message.

Applications of the A360 Bot Deployment API in Workflow Scenarios

The A360 Bot deployment API offers immense value in various workflow scenarios by enabling the deployment and integration of RPA bots. Here are some examples of workflows where the API can be utilized to generate significant benefits [13].

Invoice Processing

In organizations that handle a large volume of invoices, the A360 Bot deployment API can be employed to integrate RPA bots with the existing invoice processing workflow. The bots can automatically extract relevant data from invoices, perform validation checks, update financial systems, and generate reports. This integration reduces manual effort, minimizes errors, and accelerates the entire invoice processing cycle.

Customer Onboarding

Streamlining the customer onboarding process is crucial for businesses. By leveraging the A360 Bot deployment API, RPA bots can be integrated into the customer onboarding workflow. These bots can automate tasks such as data entry, document verification, CRM updates, and account setup. This integration enhances efficiency, improves data accuracy, and enables faster customer onboarding.

Data Migration

When organizations undergo data migration from legacy systems to new platforms, the A360 Bot deployment API can be utilized to deploy RPA bots that automate the data migration process. Bots can extract data from the legacy systems, transform it according to the new system's requirements, and seamlessly load it into the target system. This integration ensures accurate data migration, reduces manual errors, and expedites the migration timeline.

HR Processes

Human resources departments can benefit from integrating RPA bots using the A360 Bot deployment API. Bots can automate repetitive HR tasks such as employee onboarding, leave requests, performance reviews, and payroll processing. By integrating bots into HR workflows, organizations can save time, reduce administrative burden, improve data accuracy, and enhance employee satisfaction.

Supply Chain Management

In supply chain management, the A360 Bot deployment API can be utilized to deploy RPA bots

© 2024 Sai Madhur Potturu.

that automate various processes. Bots can handle tasks such as order processing, inventory management, vendor communication, and shipment tracking. Integrating bots into the supply chain workflow leads to increased operational efficiency, improved accuracy, and faster response times.

Benefits of the Solution

Streamlined Workflow Integration

The A360 Bot Deploy API allows organizations to seamlessly integrate RPA bots into their existing workflows and processes. This integration enables the automation of various steps within workflows, reducing manual intervention and improving overall process efficiency [7][14][15].

Enhanced RPA Adoption

By connecting Automation Anywhere bots with different systems and applications, organizations can take their RPA adoption to the next level. The A360 Bot Deploy API facilitates the easy deployment and management of bots within the A360 platform, providing a reliable and user-friendly solution for building automation solutions.

Improved Productivity

Integrating RPA bots into various workflows automates repetitive and rule-based tasks, freeing up employees' time to focus on more strategic and creative aspects of their work. This results in improved productivity across the organization.

Increased Accuracy

Automation of tasks through RPA reduces the likelihood of human errors, leading to improved data accuracy and quality. This is especially beneficial in critical processes like financial data handling and compliance.

Scalability and Flexibility

The A360 Bot Deploy API offers scalability, allowing organizations to deploy and manage multiple bots as per their requirements. It also provides flexibility to adapt automation solutions to evolving business needs.

Enhanced Customer and Employee Experience

RPA-driven automation improves the customer experience by reducing processing times and improving accuracy in customer-facing processes. Additionally, employees benefit from reduced mundane tasks, leading to increased job satisfaction and focus on higher-value activities.

Conclusion

In summary, this research paper provides a comprehensive solution for the smooth integration of Automation Anywhere (RPA) bots into various workflows and processes within organizations, utilizing the A360 Bot Deploy API. The integration of Automation Anywhere with established applications opens new possibilities for organizations to elevate their RPA adoption. The paper explores two distinct approaches to triggering RPA bots using the A360 Bot Deploy API: one involving passing inputs to the bot and the other without passing inputs. Both methods highlight the advantages of achieving seamless integration between workflow and RPA applications. Throughout the study, the research paper underscores the significance of harnessing the potential of the A360 Bot Deploy API to achieve seamless integration between RPA bots and organizational workflows. By embracing RPA along with this integration approach, organizations gain the power to optimize their processes, enhance productivity, and embrace digital transformation.

Declarations

Conflict of Interest Statement

The author, Sai Madhur Potturu, is employed at Zoetis Inc., specifically in the Robotics Center of Excellence (CoE) department. Zoetis Inc. is a company that provides animal healthcare products and services. The development and implementation of the digital solution presented in this manuscript align with the author's role and responsibilities within the organization. The author declares no financial or personal relationships that may have influenced the content or findings presented in this manuscript.

Data Availability Statement

The data used to support the findings of this study are available from the corresponding author upon reasonable request. The data include the PowerApps application design, RPA solution implementation details, and relevant datasets used for testing and evaluation. Access to the data will be provided to researchers or individuals to replicate the study findings or conduct further analyses related to the presented digital solution.

© 2024 Sai Madhur Potturu.

References

1. Automation Anywhere (n.d.). Automation 360. Docs.Automationanywhere.com.
2. Mahey, H. (2020). Robotic Process Automation with Automation Anywhere: Techniques to fuel business productivity and intelligent automation using RPA. *Packt Publishing Ltd.*
3. Anagnoste, Sorin. (2017). Robotic Automation Process - The next major revolution in terms of back-office operations improvement. *Proceedings of the International Conference on Business Excellence*, vol.11(1):676-686.
4. Automation Anyhwere (n.d.). Process Orchestration and Automation—A Winning Combination for Business.
5. [Automation Anywhere]. (2020). How to Trigger a Bot from the Control Room API | #AAIllustrates with Micah Smith Episode 15 [Video]. Youtube.com.
6. Automation Anyhwere (n.d.). Integrations. Docs.Automationanywhere.com/.
7. Automation Anywhere (n.d.). Bot deploy API. Docs.Automationanywhere.com.
8. Automation Anywhere (n.d.). Control Room APIs. Docs.Automationanywhere.com.
9. Automation Anywhere (n.d.). Bot editor for creating bots. Docs.Automationanywhere.com.
10. Automation Anywhere (n.d.). Authentication API. Docs.Automationanywhere.com.
11. Automation Anywhere (n.d.). Multi-login user. Docs.Automationanywhere.com.
12. Automation Anywhere (n.d.). Your variables (user-defined). Docs.Automationanywhere.com.
13. beachnet (n.d.). The Benefits of Automation for Different Industries.
14. infomentum (n.d.). API+RPA.
15. (n.d.). 7 Biggest Benefits of RPA (Robotic Process Automation).

Cite this article: Sai M. Potturu (2024). Seamless Integration of Automation Anywhere (RPA) Bots into Organizational Systems and Workflows using the A360 Bot Deploy API. *Scientific Research and Reports*, BioRes Scientia Publishers. 1(1); DOI: 10.59657/2996-8550.brs.24.001

Copyright: © 2024 Sai Madhur Potturu, this is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Article History: **Received:** December 12, 2023 | **Accepted:** December 27, 2023 | **Published:** January 04, 2024