UNIT-V

Email Automation & Exceptional Handling: Email Automation, Incoming Email automation, Sending Email automation, Debugging and Exception Handling, Debugging Tools, Strategies for solving issues, Catching errors.

Email Automation :

Email automation in UiPath involves using the UiPath platform to automate various email-related tasks, such as sending emails, reading emails, and processing email attachments. UiPath provides a set of activities and components that make it easier to work with email systems like Microsoft Outlook, Gmail, and others. Below are the steps to set up email automation in UiPath:

- Install UiPath Studio: Ensure that you have UiPath Studio installed on your computer. You can download it from the UiPath website.
- Open a New Project: Launch UiPath Studio and create a new project where you will develop your email automation workflow.
- Add Email Activities: To work with emails, you need to use the "Email" activities provided by UiPath. These activities can be found in the "UiPath.Mail.Activities" package. You can add this package by going to the "Manage Packages" section in UiPath and searching for "UiPath.Mail.Activities."
- Configure Email Account: Before using email activities, you need to configure your email account in UiPath. This can be done through the "Account" pane in the "Mail" tab. You will need to provide the email address, password, and other relevant details for the email account you want to automate.
- Use Email Activities: Once your email account is configured, you can use activities like:
    - Send Outlook Mail Message: Use this activity to send emails from your configured email account.
    - Get Outlook Mail Messages: Use this activity to retrieve emails from your inbox or other folders.
    - Save Attachments: This activity can be used to save email attachments to a specified folder.
    - Move Outlook Mail Message: Move emails to different folders.
- Delete Outlook Mail Message: Delete unwanted emails.
- Automate Email Processing: You can automate various tasks like reading specific emails, extracting information from them, and taking actions based on email content. For example, you can create a workflow that automatically responds to emails with specific keywords or moves emails to different folders based on their content.
- Error Handling: Implement error handling mechanisms in your workflow to handle exceptions that may occur during email automation, such as network issues or email server problems.
- Testing and Debugging: Thoroughly test your email automation workflow in UiPath to ensure it performs as expected. Use the debugging features in UiPath Studio to troubleshoot any issues.

- Scheduling: If you want to run your email automation workflow at specific times or intervals, you can use UiPath Orchestrator to schedule and manage your automation jobs.
- Deployment: Once your email automation workflow is complete and tested, you can deploy it to a UiPath Robot or Orchestrator for production use.

Remember to handle sensitive information like passwords securely when automating email tasks, and follow best practices for automation and email communication to ensure the security and reliability of your email automation processes.

Incoming Email automation

Incoming email automation in UiPath involves automating processes that respond to emails received in your inbox or other email folders. You can use UiPath to perform actions such as reading incoming emails, extracting information from them, and triggering specific workflows based on the content of the emails. Here's a step-by-step guide to setting up incoming email automation in UiPath:

- Install UiPath Studio: Ensure that you have UiPath Studio installed on your computer. You can download it from the UiPath website.
- Open a New Project: Launch UiPath Studio and create a new project where you will develop your incoming email automation workflow.
- Add Email Activities: To work with incoming emails, you need to use the "Email" activities provided by UiPath. These activities can be found in the "UiPath.Mail.Activities" package. You can add this package by going to the "Manage Packages" section in UiPath and searching for "UiPath.Mail.Activities."
- Configure Email Account: Before using email activities to automate incoming emails, you need to configure your email account in UiPath. This can be done through the "Account" pane in the "Mail" tab. You will need to provide the email address, password, and other relevant details for the email account you want to monitor for incoming emails.
- Use Email Activities for Incoming Emails:
    - Get Outlook Mail Messages or Get IMAP Mail Messages: Use one of these activities (depending on your email provider) to retrieve incoming emails from your inbox or other folders. Configure the activity with appropriate filtering options such as subject, sender, or date.
- Process Incoming Emails: After retrieving the incoming emails, you can process them using various UiPath activities. Some common tasks include:
    - Iterate Through Emails: Use a "For Each" loop to iterate through the list of retrieved emails.
    - Extract Information: Use activities like "Get IMAP Mail Message" to extract details from the emails, such as the subject, sender, body, attachments, and any specific data you need.
    - Conditional Logic: Based on the content or characteristics of the incoming emails, you can implement conditional logic to trigger specific actions or workflows.

- Automation Logic: Implement the specific automation logic that needs to be triggered based on the content of the incoming emails. For example:
    - Respond to customer inquiries or support requests.
    - Update databases or perform data entry based on information in the emails.
    - Forward emails to relevant team members or departments.
    - Generate reports or notifications based on email data.
    - Error Handling: Implement error handling mechanisms in your workflow to handle exceptions that may occur during email processing, such as network issues or email server problems.
- Testing and Debugging: Thoroughly test your incoming email automation workflow in UiPath to ensure it performs as expected. Use the debugging features in UiPath Studio to troubleshoot any issues.
- Scheduling (Optional): If you want your automation to run continuously, you can schedule your workflow to check for incoming emails at specific intervals using UiPath Orchestrator.
- Deployment: Once your incoming email automation workflow is complete and tested, you can deploy it to a UiPath Robot or Orchestrator for production use.

Ensure that you follow best practices for email automation, handle sensitive information securely, and consider email security and privacy regulations when implementing incoming email automation processes.

Sending Email Automation :
Sending email automation in UiPath involves automating the process of composing and sending emails using your email client or server. You can use UiPath to send emails for various purposes, such as sending notifications, reports, or responses to specific events or triggers. Here's a step-by-step guide to setting up email automation for sending emails in UiPath:

- Install UiPath Studio: Ensure that you have UiPath Studio installed on your computer. You can download it from the UiPath website.
- Open a New Project: Launch UiPath Studio and create a new project where you will develop your email sending automation workflow.
- Add Email Activities: To send emails, you need to use the "Email" activities provided by UiPath. These activities can be found in the "UiPath.Mail.Activities" package. You can add this package by going to the "Manage Packages" section in UiPath and searching for "UiPath.Mail.Activities."
- Configure Email Account: Before using email activities to automate sending emails, you need to configure your email account in UiPath. This can be done through the "Account" pane in the "Mail" tab. You will need to provide the email address, password, and other relevant details for the email account you want to use for sending emails.
- Use Email Activities for Sending Emails:
    - Send Outlook Mail Message or Send SMTP Mail Message: Depending on your email client or server, you can use either of these activities to

compose and send emails.
    - Send Outlook Mail Message: Use this activity if you are using Microsoft Outlook as your email client.
    - Send SMTP Mail Message: Use this activity if you want to send emails using an SMTP server. You will need to configure the SMTP server settings in this activity.
- Compose the Email: Use the chosen email activity to compose the email you want to send. Configure the following parameters:
    - Recipient(s): Specify the email address(es) of the recipient(s).
    - Subject: Provide the subject of the email.
    - Body: Enter the content of the email, which can include text, HTML, or a combination of both.
    - Attachments: Attach any files that you want to include with the email.
- Automation Logic: Implement the specific automation logic that triggers the email sending. For example:
    - Send automated reports at specific times.
    - Notify stakeholders when certain conditions are met in a workflow.
    - Send confirmation emails for completed processes.
- Error Handling: Implement error handling mechanisms in your workflow to handle exceptions that may occur during email sending, such as network issues or authentication problems.
- Testing and Debugging: Thoroughly test your email sending automation workflow in UiPath to ensure it performs as expected. Use the debugging features in UiPath Studio to troubleshoot any issues.
- Scheduling (Optional): If you want your email sending automation to occur at specific times or intervals, you can schedule your workflow using UiPath Orchestrator.
- Deployment: Once your email sending automation workflow is complete and tested, you can deploy it to a UiPath Robot or Orchestrator for production use.

Ensure that you follow best practices for email automation, handle sensitive information securely (such as email credentials), and consider email security and privacy regulations when implementing email sending automation processes.

Debugging and Exception Handling

Debugging and exception handling are crucial aspects of UiPath automation workflows. Properly handling exceptions and debugging your workflows helps ensure that your automation processes run smoothly and reliably. Here's a guide on debugging and exception handling in UiPath:

Debugging in UiPath:

Set Breakpoints:

- To start debugging, place breakpoints in your workflow. You can do this by clicking on the activity where you want to pause execution and selecting "Toggle Breakpoint" from the context menu.

Run in Debug Mode:

- Click the "Debug File" button or press F5 to run your workflow in debug mode. The workflow will execute until it reaches a breakpoint.

Inspect Variables and Output:

- While debugging, you can inspect the values of variables by hovering over them in the Locals panel or by adding them to the Watch panel.
- Use the Output panel to view log messages and execution details.

Step Through Execution:

- Use the debugging controls, such as "Step Into" (F11), "Step Over" (F10), and "Step Out" (Shift + F11), to navigate through your workflow one activity at a time.
- This allows you to closely monitor the execution and identify issues.

Break and Resume:

- You can pause the execution at any point by clicking the "Pause" button. To resume execution, click "Continue."

View Call Stack:

- The Call Stack panel shows the hierarchy of activities and their execution order, which can be helpful for understanding the flow of your workflow.

Exception Handling in UiPath:

Exception handling is essential for gracefully managing errors and exceptions that may occur during automation:

Try-Catch Activity:

- Use the "Try Catch" activity to enclose activities that might generate exceptions.
- Inside the "Try" block, place the activities that you want to monitor for exceptions.

Catch Blocks:

- Add one or more "Catch" blocks inside the "Try Catch" activity to handle specific exceptions.
- You can catch exceptions of different types (e.g., System.Exception, System.IO.IOException) and define actions to take when each exception occurs.

Finally Block:

- Optionally, you can include a "Finally" block to specify actions that should always occur, whether an exception occurs or not.

Logging Exceptions:

- Inside the "Catch" blocks, use the "Log Message" activity to log details about the exception, including the exception message and relevant information.

Rethrowing Exceptions:

- In some cases, you may want to rethrow exceptions after logging them to ensure that higher-level exception handling can address the issue.

Global Exception Handler (Optional):

- UiPath allows you to set up a global exception handler in the project settings. This handler can capture unhandled exceptions and take predefined actions, such as sending notifications.

Retry Mechanisms:

- Use the "Retry Scope" activity to implement retry mechanisms for specific activities. This helps handle transient errors by retrying the activity a specified number of times.

Exception Types:

- UiPath provides predefined exception types, but you can also create custom exceptions to handle specific scenarios.

Test Exception Handling:

- Thoroughly test your exception handling logic by intentionally causing exceptions to ensure that your workflow handles them appropriately.

By effectively using debugging and exception handling in UiPath, you can create robust and reliable automation workflows that can handle unexpected situations and recover gracefully from errors.

Debugging Tools:

UiPath provides several debugging tools and features within UiPath Studio to help you identify and resolve issues in your automation workflows. These tools are essential for troubleshooting and improving the reliability of your automated processes. Here are some of the key debugging tools in UiPath:

Breakpoints:

- Breakpoints allow you to pause the execution of your workflow at specific points to inspect variables, check the flow, and identify issues.
- You can set breakpoints by right-clicking on an activity and selecting "Toggle Breakpoint" or by pressing F9 when the activity is selected.

Step Into (F11):

- Use the "Step Into" debugging command to dive into the execution of an activity. It will take you inside the activity if it has child activities.
- This command helps you understand the details of how an activity works.

Step Over (F10):

- The "Step Over" command allows you to execute the current activity and move to the next one without diving into child activities.
- It's useful for quickly advancing through the workflow.

Step Out (Shift + F11):

- When you're inside a child activity, the "Step Out" command takes you back to the parent activity.

- This command is helpful for moving up the execution hierarchy.

Pause and Continue:

- While debugging, you can pause the execution at any time by clicking the "Pause" button or pressing Shift + F5. To continue execution, click "Continue" or press F5.

Locals Panel:

- The Locals panel displays information about variables and their values at the current point in the workflow.
- You can hover over variables to view their values or add them to the Watch panel for monitoring.

Watch Panel:

- The Watch panel allows you to add specific variables you want to monitor during debugging.
- You can add variables by right-clicking and selecting "Add Watch."

Output Panel:

- The Output panel displays log messages and provides information about workflow execution.
- It's helpful for understanding the sequence of activities and identifying issues.

Call Stack Panel:

- The Call Stack panel shows the hierarchy of activities and their execution order, helping you understand the flow of your workflow.
- It's particularly useful when you have nested activities.

Interactive Debugging:

- UiPath Studio supports interactive debugging, allowing you to interact with applications during debugging.
- This is beneficial when you need to inspect or manipulate elements within an application while debugging.

Exception Handling:

- Implement proper exception handling in your workflows using the "Try Catch" activity to capture and manage exceptions effectively.
- Log exceptions and take appropriate actions based on the type of exception.

Break and Stop:

- The "Break" button interrupts the debugging session, and you can examine the state of variables and activities.
- The "Stop" button terminates the debugging session altogether.

These debugging tools in UiPath Studio are essential for troubleshooting issues, verifying the correctness of your automation logic, and ensuring the reliability of your automated processes. By utilizing these tools effectively, you can identify and resolve

errors, optimize your workflows, and improve the overall performance of your automation projects.

Strategies for Solving Issues:

Solving issues in UiPath, whether they are related to your automation workflows, activities, or platform setup, requires a systematic and strategic approach. Here are some strategies to help you effectively resolve issues in UiPath:

Identify and Understand the Issue:

- Start by clearly identifying the issue you're facing. Understand what is not working as expected and gather as much information as possible about the problem.
- Check error messages, logs, and any available diagnostic information.

Reproduce the Issue:

- Attempt to reproduce the issue in a controlled environment. If you can consistently replicate the problem, it will be easier to debug and find a solution.

Check for Known Issues:

- Consult UiPath's official documentation, release notes, and community forums to check if the issue you're experiencing is a known bug or limitation. UiPath often provides solutions or workarounds for known issues.

Review Workflow Logic:

- If the issue is related to your automation workflow, review the logic of your workflow. Check variables, conditions, and activities to ensure they are correctly configured.
- Use the debugging tools in UiPath Studio to step through your workflow and identify where the issue occurs.

Examine Input Data:

- If your automation relies on input data, review and validate the input data to ensure it is accurate and complete.
- Check for data format issues, missing values, or inconsistencies.

Activity Properties:

- Verify that the properties of activities are set correctly. Pay attention to input arguments, selectors, and other parameters.
- Ensure that activity dependencies are resolved, such as package versions and external resources.

Log and Debug:

- Implement proper logging in your workflows using the "Log Message" activity. This helps capture relevant information during execution.

- Use the debugging tools in UiPath Studio to step through your workflow, inspect variables, and identify the source of the problem.

Exception Handling:

- Implement robust exception handling using the "Try Catch" activity to gracefully handle errors and exceptions. Log detailed information about exceptions.
- Ensure that your exception handling logic covers various error scenarios.

Consult UiPath Community:

- If you're unable to resolve the issue on your own, consider posting your problem on the UiPath Community forum. The UiPath community is active and often provides helpful solutions and insights.

Update or Reinstall Packages:

- If you suspect that the issue is related to specific activities or packages, try updating or reinstalling them through the "Manage Packages" feature in UiPath Studio.

Check System Requirements:

- Ensure that your system meets the hardware and software requirements for running UiPath. Inadequate system resources can lead to performance issues.

Network and Security:

- Verify that your network connectivity and security settings are not causing the issue. Firewalls, proxies, and network configurations can impact automation.

UiPath Version:

- Ensure that you are using a supported version of UiPath. Consider upgrading to the latest stable version if you are experiencing compatibility issues.

Documentation and Training:

- Take advantage of UiPath's official documentation and training resources. Sometimes, issues arise from a lack of understanding or familiarity with the platform.

Seek Professional Help:

- If the issue persists and is critical to your business operations, consider contacting UiPath's support services or engaging a UiPath certified professional for assistance.

Remember that effective issue resolution often involves a combination of technical troubleshooting, logical analysis, and collaboration with the UiPath community and support resources. Document your findings, solutions, and best practices for future reference, as this can help streamline issue resolution in the long term.

Catching errors:

Catching and handling errors in UiPath is essential for creating robust and reliable automation workflows. You can use the "Try Catch" activity to gracefully capture and manage errors that may occur during the execution of your processes. Here's how to catch errors in UiPath:

Add the "Try Catch" Activity:

- Drag and drop the "Try Catch" activity from the activities panel into your workflow. This activity allows you to enclose a section of your workflow where you anticipate errors may occur.

Configure the "Try" Block:

- Inside the "Try Catch" activity, place the activities that you want to monitor for exceptions in the "Try" block. These are the activities that you expect may generate errors.

Configure "Catch" Blocks:

- Within the "Try Catch" activity, you can add one or more "Catch" blocks to handle specific types of exceptions.
- To add a "Catch" block, click the "Add Catch" button and select the exception type you want to handle. You can specify the exception type, like System.Exception, or use specific exceptions like System.IO.IOException.
- For each "Catch" block, define the actions you want to take when the specified exception occurs. These actions can include logging, notification, or custom handling logic.

Use a "Finally" Block (Optional):

- Optionally, you can add a "Finally" block within the "Try Catch" activity. The activities inside the "Finally" block will execute regardless of whether an exception occurs or not.
- You can use the "Finally" block to perform cleanup tasks or actions that should always occur.

Log Exceptions:

- Inside the "Catch" blocks, it's a good practice to use the "Log Message" activity to log details about the exception. Include information such as the exception message, timestamp, and any relevant context.
- Logging exceptions provides valuable information for debugging and auditing purposes.

Rethrow Exceptions (Optional):

- In some cases, you may want to rethrow the exception after handling it in a "Catch" block. Rethrowing allows higher-level exception handlers to address the issue.
- To rethrow an exception, use the "Throw" activity within a "Catch" block.

Here's a basic example of how a "Try Catch" activity might be structured:
plaintext

```
Try
    // Activities that may generate exceptions go here
Catch System.Exception
    // Handle the exception (e.g., log, notify, or recover)
    Log Message: "An exception occurred: " + Exception.Message
Finally
    // Cleanup or actions that always execute
End Try
```

By using the "Try Catch" activity effectively, you can ensure that your automation processes can gracefully handle errors and exceptions, preventing them from causing workflow failures. It also allows you to log information for troubleshooting and implement appropriate actions to recover from errors.