**Package execution process:**

**Step 01:**

* **Sql drop org table:**

The SQL code checks for existence of two table (“org” and “ Day Jobs Master Logs”) in the “dbo” schema. If either or both tables exist, they are dropped from the data base. This is typically used before recreating these tables with a new structure or data.

**Step 02:**

* **Scr get portfolioID from ApplicationName**:

The code seems to be extracting a portion of the string (likely an application name) and storing it in a variable named “PortfolioID”. This “PortfolioID” is than used for the further processing within the SSIS package.

* The code contains some commented-out lines (lines starting with a single quote '), which are not executed but are used for documentation or debugging purposes.
* The code includes a MsgBox call, which would display a message box with the extracted "PortfolioID" if uncommented. This is likely for debugging or testing purposes during development.

**Overall Function:**

In essence, this code snippet extracts a value from a variable named wApplicaitonName and stores it in an SSIS variable named "PortfolioID." It then logs an informational message and sets the script task result to "Success." This suggests that the script is likely part of a larger workflow within the SSIS pac

kage, where the extracted "PortfolioID" will be used in subsequent tasks, potentially for tasks related to email generation or data processing.

**Step 03:**

**dft get email parameters:**

This code is to process email parameters extracted from a data source and prepare them for sending an email within an SSIS package.

Key Functionality:

**1. Parameter Processing:**

* The code iterates through rows of data received from a data source (likely an OLE DB Source).
* For each row, it checks if the current row contains values for email parameters like "To," "From," "Subject," and others.
* If a value is found, it assigns it to the corresponding script variable (gEmailTo, gEmailFrom, gEmailSubjectLine, etc.).

**2. Environment Check & Subject Line Modification:**

* It checks the current environment (likely using an SSIS variable).
* If the environment is not "PRD" (production), it appends the environment name to the gEmailSubjectLine.

**3. Message Construction:**

* It constructs the email message body based on the available parameters. In this case, it seems to be a simple message indicating that processing has completed for a specific PortfolioID.

**4. Error Handling:**

* It checks for missing required parameters (To, From, Subject). If any of these are missing, it triggers an error event within the SSIS package.

**5. Variable Assignment:**

* Finally, it assigns the processed email parameters (e.g., gEmailFrom, gEmailSubjectLine) to SSIS variables, making them available for subsequent tasks in the package (such as the Send Mail task).

**6. Code Structure:**

It uses the Overrides keyword to override methods inherited from the base class, allowing for custom logic within the Script Task.

Overall, this code demonstrates a basic approach to processing and preparing email parameters within an SSIS Script Task. It can be further enhanced with features like dynamic recipient lists, attachments, HTML formatting, and more robust error handling.

**Step 04:**

* **Sql\_Call\_DayJobMaster:**

This Execute SQL Task is designed to execute the stored procedure "DayJobsMaster" using the specified OLE DB connection.

* **Scr override email variables:**

This code retrieves email subject and body information from SSIS variables and prepares them for further processing. The commented-out message box code suggests that it might have been used for testing or debugging purposes during development.

**Key Functionality:**

* **Variable Assignment:**

The code assigns the value of the e\_mail\_subject SSIS variable to the local variable sEmailSubjectLine. \* It assigns the value of the XML\_e\_mail\_body\_log SSIS variable to the local variables EmailMessage.

* **Message Display (Commented Out):**

The commented-out lines suggest that the code was originally intended to display the value of sEmailMessage in a message box. This is likely for debugging or testing purposes.

* **Setting Task Result:**

The code sets the Dts.TaskResult to ScriptResults.Success to indicate successful execution of the script task.

* **Sql load runtime e\_mail:**

This executes SQL task place a crucial role in extracting the Email\_body data from the data base.

* **ScrMailRun Times:**

The primary purpose of this code is to send an email based on variables and configurations within the SSIS package.

Key Functionality:

**1. Variable Initialization:**

The code declares and initializes several variables to store email-related information:

* e\_mail\_to\_TEST: A variable to store the email address based on the environment.
* EmailTo: The actual recipient's email address.
* EmailFrom: The email address of the sender.
* EmailSubjectLine: The subject of the email.
* Environment: The current environment (e.g., "DEV," "TEST," "PRD").
* EmailMessage: The body of the email.
* EmailConnectionString: The connection string for the SMTP server.

**2. Environment Check and Recipient Determination:**

* The code checks the value of the Environment variable.
* If the environment is "PRD" (production), it uses the value of the e\_mail\_to variable for the recipient.
* If the environment is not "PRD," it sets the recipient to a hardcoded email address ("[roberto.perez@molinahealthcare.com](mailto:roberto.perez@molinahealthcare.com)").

**3. Email Creation:**

* The code creates a new MailMessage object with the following properties:
* From: The value of EmailFrom.
* To: The value of EmailTo.
* Subject: The value of EmailSubjectLine concatenated with the environment (if not "PRD").
* Body: The value of EmailMessage.

**4. SMTP Client Configuration:**

* The code creates a new SmtpClient object using the EmailConnectionString variable.
* It sets the IsBodyHtml property of the MailMessage to True (assuming the email message is in HTML format).
* It adds a recipient address to the MailMessage object (this might be redundant if EmailTo is already set correctly).

**5. Sending the Email:**

The code calls the Send() method of the SmtpClient object to send the email.

**6. Setting Task Result:**

* The code sets the Dts.TaskResult to ScriptResults.Success to indicate successful script execution.

This VBScript code demonstrates a basic email sending mechanism within an SSIS Script Task. It retrieves necessary information from SSIS variables, creates an email message object, and sends the email using an SMTP client.

**Sql load org table:**

This SQL query appears to be designed to retrieve information about organizations and their associated members, along with email distribution details. It likely aims to generate a report or data set for further analysis or processing.

**Key Elements:**

● **Data Retrieval:** The query selects several columns:

○ O.OrgType: The type of organization.

○ O.OrgID: The unique identifier for the organization.

○ ppt.ProviderName + ' - ' + psl.ProviderName as OrgName: The combined name of the provider and service location.

○ COUNT(mp.MemberID) as MemberPanelCount: The total number of members associated with the organization.

○ vms.dbo.GetOrgEmailListActive(O.OrgID) as EmailDistribution: This likely calls a function (vms.dbo.GetOrgEmailListActive) to retrieve the active email distribution list for the organization.

● **Table Joins:** The query uses multiple joins to combine data from different tables:

○ dbo.ProviderPayTo

○ dbo.ProviderServiceLocation

○ dbo.Organization

○ dbo.MemberPanel

○ dbo.MemberPanelAttestation

○ dbo.OrganizationAttestation

● **Filtering Conditions:**

○ where O.Active = 1: Filters the results to include only active organizations.

○ @PanelDate between mp.EffectiveDate and mp.TerminationDate: Filters members based on their effective and termination dates within a specific panel date range.

○ COUNT(mp.MemberID) > 0: Filters out organizations with no members.

○ COUNT(mpa.MemberPanelID) = 0: This condition is unclear without further context. It might filter out organizations that don't meet specific attestation criteria.

● **Grouping and Ordering:**

○ group by O.OrgType, O.OrgID, ppt.ProviderName, psl.ProviderName: Groups the results by organization type, ID, and provider/service location.

○ having COUNT(mp.MemberID) > 0: Filters out groups with zero members.

○ order by 2: Orders the results based on the OrgID column.

● **Variable Declaration:**

○ The query declares a variable @ProcStartTime to store the start time of the procedure.

○ It also declares a variable @PanelDate to represent a specific date, likely used for filtering purposes.

**Sql extract org foreach loop:**

This SQL query retrieves essential information for sending email reminders to organizations that haven't completed their attestations. It selects organizational details, constructs a standardized email subject and body, and retrieves the active email distribution list for each organization. The query then filters out organizations without active email lists.

**Key Elements:**

● **Data Retrieval:** The query selects the following columns:

○ o.OrgType: The type of organization.

○ o.OrgID: The unique identifier for the organization.

○ o.OrgName: The name of the organization.

○ o.MemberPanelCount: The number of members associated with the organization.

○ 'VMS Attestation Reminder' as email\_subject: A constant string for the email subject.

○ <html><body>...</body></html> as e\_mail\_body: An HTML formatted email body with placeholders for organization name and a message regarding the need for attestation.

○ dbo.GetOrgEmailListActive\_3052(o.OrgID) as EmailList: This likely calls a function (dbo.GetOrgEmailListActive\_3052) to retrieve the active email distribution list for the organization.

● **Table Join:** The query joins the dbo.org table with the dbo.emailnotice table. The join condition is based on EmailNotice being not null. This suggests that the emailnotice table holds information about email notifications or reminders.

● Filtering Condition:

○ WHERE dbo.GetOrgEmailListActive\_3052(o.OrgID) is not NULL: This condition filters the results to include only organizations that have an active email distribution list.

**Fel loop through e-mail script:**

**ScrMailAttestationNotice:**

The code demonstrates a basic email sending functionality. It could be enhanced by adding error handling, logging, and more robust configuration management.

● It creates a MailMessage object with the sender, recipient, subject, and message body.

● It configures an SmtpClient using connection details.

● It sets the email body as HTML.

● It adds the recipient to the email.

● It sends the email using the SmtpClient.

● Finally, it logs the email parameters.

**Dft\_create\_summary\_file:**

This task is responsible for extracting data from an OLE DB source, transforming it, and loading it into a flat file.

**1. Data Extraction:**

● The task connects to an OLE DB source using the specified connection details and retrieves data from the specified table or view.

**2. Data Transformation:**

● The data undergoes transformations using derived columns.

○ Specifically, the "OrgName" column is modified using the expression REPLACE(OrgName, HEX(09), ''). This likely removes a specific character (potentially a tab) from the "OrgName" field.

**3. Data Loading:**

● The transformed data is then loaded into a flat file.

○ The "Flat File Destination Editor" configures the connection to the target file, including options like overwriting existing data.

**scrMailSummaryFile:**

This task is responsible for sending an email notification. It retrieves email-related information from DTS variables, creates an email message with the specified content and attachment, and then sends it using an SMTP server.

**Functionality:**

1. **Retrieves Email Information:** It reads values for email recipients (EmailTo), sender (EmailFrom), subject (EmailSubjectLine), message body (EmailMessage), connection string (EmailConnectionString), and attachment path (EmailAttachment) from DTS variables.

2. **Creates Email Object:** It creates a new MailMessage object using the retrieved information. The subject line is dynamically constructed by concatenating the EmailSubjectLine with the Environment name.

3. **Adds Attachment:** It creates an Attachment object using the path from the EmailAttachment variable and adds it to the email message.

4. **Sends Email:** It uses an SmtpClient object with the EmailConnectionString to send the email.

5. **Sets Task Result:** Finally, it sets the Dts.TaskResult to ScriptResults.Success to indicate successful email delivery.