**ReadEmail()Method**

* **Variables Initialization:** Initializes various variables to manage email reading and processing. These include flags (isServerConnected, success, saveAttachment, saveEmail), counters (iEmailLoop, iTotalEMailsInMailbox, iSrNo, iTotalEmailAttachements), and arrays (saUnseenUidls, saSeenUidls, saUidls, fetchEmail).
* **Data Extraction:** Extracts necessary data from the DataRow passed to the method (dataRow). This includes rowUid, clientId, companyId, lastMailReadTime, saveAttachment, saveEmail, and fetchEmail.
* **Chilkat Initialization:** Initializes Chilkat components for POP3 (mailBoxPop3) and IMAP (mailBoxImap) mail handling.
* **Internet Connection Check:** Uses CheckConnection.IsConnectedToInternet() method to verify internet connectivity.
* **Server Type Identification:** Retrieves the server type (POP3 or IMAP) from dataRow.
* **Mail Server Connection:** Depending on the serverType :
  + **IMAP Connection:** Unlocks and connects to the IMAP server (mailBoxImap), sets parameters such as port and login credentials.
  + **POP3 Connection:** Unlocks and connects to the POP3 server (mailBoxPop3), sets parameters such as server host, SSL usage, port, and establishes connection.
* **Connection Status:** Verifies if the connection to the server (isServerConnected) is successful.

1. **Initial Server Check:**

* Checks if the server (POP3 or IMAP) is connected (isServerConnected).

1. **POP3 Server Handling:**

* If the server type is POP3, it
  + Logs the connection.
  + Creates directories if they don't exist for storing seen UIDs and attachments.
  + Loads previously seen UIDs from a file and retrieves current UIDs from the POP3 mailbox.
  + Checks for unread emails, stores unseen UIDs, and logs the count of unread emails.

1. **IMAP Server Handling:**

* If the server type is IMAP, it:
  + Logs the connection.
  + Selects the Inbox mailbox.
  + Searches for emails since a specified date (lastMailReadTime).
  + Retrieves the unseen message set and logs the count of fetched emails.

1. **Email Processing Loop:**

* Loops through each unread email (iEmailLoop):
  + Fetches the email based on server type (POP3 or IMAP).
  + Creates directories for saving email attachments.
  + Checks if the email ID already exists in the database.
  + Saves the email content and attachments if conditions are met.
  + Processes specific attachment types (PDF, PRN, ACK, DMS\_) and updates database records accordingly.
  + Updates the last read time of emails.

1. **Save Seen UIDs:**

* Saves the seen UIDs back to a file after processing.

**Now explain the code for save files.**

1. **FileInfo Initialization:**

* FileInfo fileInfo = new FileInfo(mailEmailMessage.GetAttachmentFilename(i));
  + This line creates a FileInfo object named fileInfo by fetching the filename of the attachment at index i from mailEmailMessage.

1. **File Extension Extraction:**

* string filExtension = fileInfo.Extension;
  + Extracts the file extension (including the dot) of the attachment file using fileInfo.Extension.

1. **File Path Creation:**

* string filePath = mailFolder + "\\" + clientId + "\_" + companyId + "\\Email\\Attachment\\" + sEmailUniqueId + "\\" + mailEmailMessage.GetAttachmentFilename(i);
  + Constructs the complete file path where the attachment will be saved. Here’s how it is structured:
    - mailFolder: Root directory path for saving emails.
    - clientId and companyId: Unique identifiers for the client and company.
    - sEmailUniqueId: Unique identifier for the email.
    - mailEmailMessage.GetAttachmentFilename(i): Original filename of the attachment.

Example: If mailFolder is "C:\\Emails", clientId is "Client1", companyId is "Company1", sEmailUniqueId is "UniqueEmail123", and the attachment filename is "document.pdf", then filePath would be:

C:\Emails\Client1\_Company1\Email\Attachment\UniqueEmail123\document.pdf

**Explanation of Conditions (PDF, PRN, ACK, DMS\_):**

1. **PDF File Handling:**

* Checks if the attachment file has a .pdf extension.
* If conditions (saveEmail == true and saveAttachment == true) are met
  + Checks the subject of the email (mailEmailMessage.Subject) to determine further actions (Final OOC or First copy of BOE).
  + Constructs specific paths (FinalBOE or FirstCopyBOE directories) based on the subject.
  + Saves the attachment file using mailEmailMessage.SaveAttachedFile(i, filePath)
  + Constructs an SQL query to insert attachment details into the database (EmailAttachmentMaster).
* Subject Check (if (mailEmailMessage.Subject.ToLower().Contains("final ooc")) and else if (mailEmailMessage.Subject.ToLower().Contains("first copy of boe"))):
  + Within the save conditions, it further checks the subject of the email message (mailEmailMessage.Subject) in lowercase:
    - First Check ("final ooc"):
      * If the subject contains "final ooc", it executes code to handle an "Electronic Final OOC Copy."
    - Else If Check ("first copy of boe")
      * If the subject contains "first copy of boe", it executes code to handle a "First Copy of BOE."

1. **PRN File Handling:(**BE Files**)**

* Checks if the attachment file has a .prn extension.
* Similar to PDF handling, saves the attachment file and updates the database with details.

1. **ACK File Handling: (for updating BENo or Be date)**

* Checks if the attachment file has a .ack extension.
* Retrieves attachment content as UTF-8 string using Encoding.UTF8.GetString(mailEmailMessage.GetAttachmentData(i)).
* Processes content based on conditions (fileInfo.FullName.Contains("CHCAI02") or CHCAE02), updating database records (JobHeader table) accordingly. (Based on email subject JobNumber than we update ThokaNo (BENo) or thoka date in jobheader table )

1. **DMS\_ File Handling:(ESanchit document )**

* Checks if the attachment file has a .dms\_ extension.
* Retrieves attachment content as UTF-8 string.
* Processes content to insert records into the ESanchitDocument table based on conditions (ICEGATEUserId and ImageReferenceNo).

Conclusion:

Each file attachment undergoes specific handling based on its extension and the conditions defined within the loop. The filePath variable is crucial as it dictates where and how attachments are saved and processed further based on the application's business logic.