EMAIL INTERROGATOR

MS Outlook

Objective

Create a utility to extract email senders from the mailbox. This will then be used to define rules on which emails to delete immediately or at a certain age

Result

The program allows the user to produce a delimited file containing the email subject, received date, sender name, sender email address.

In a subsequent document, this file will be used to construct a reference list that will dictate the actions that will be taken on emails in the mail box

Platform

Windows, Outlook.

Tested on Win 10, 2020

Solution

Starting with the utilities that will be used to create the output file and run log. This will make use of CreateObject("Scripting.FilesystemObject") - referencing the scripting tools provided on the Windows system.

We're going to set up the utilities first, then the Procedure that sets the configurations and lastly the main program and caller (Director)

Function confirmFolder(FilePath As String) As Boolean

```
'confirms or creates folder used by the app
Dim p() As String 'path holder
Dim i As Integer, extended_path As String

On Error GoTo eh
Set FSOobj = CreateObject("Scripting.FilesystemObject")
p() = Split(FilePath, "\")

extended_path = p(0)
For i = 1 To UBound(p) 'create folder but only 1 level at a time
   extended_path = extended_path & "\" & p(i)
   If FSOobj.FolderExists(FilePath) = False Then
      FSOobj.CreateFolder FilePath       'If path doesn't exist, create it.
   End If
Next i
confirmFolder = True
Exit Function
eh:
MsgBox ("folder could not be confirmed to start logging")
confirmFolder = False
End Function
```

Function createLogFile(LogName As String) As Boolean

```
On Error GoTo eh
Set FSOobj = CreateObject("Scripting.FilesystemObject")
If FSOobj.FileExists(LogName) = False Then
   Set a = FSOobj.CreateTextFile(LogName, True)
```

```
    a.WriteLine ("Log start")
    a.Close
End If
createLogFile = True
Exit Function
eh:
MsgBox ("Log file creation failed")
End Function
```

Sub AddLog(filename As String, Entry As String, Mode As String)

```
Const ForReading = 1, ForWriting = 2, ForAppending = 8, TristateTrue = 0 '(need -1 to use utf8, 0 = Ascii)
Dim fs, f
Dim content As String, cleaned_string As String

cleaned_string = Remove_non_ascii(Entry)

If Mode = "Log" Then
    content = Format(Now, HHMM) & " : " & cleaned_string & vbCrLf
Else
    content = cleaned_string & vbCrLf
End If

Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.OpenTextFile(filename, ForAppending, True, TristateTrue) ' last -1 is for utf8 encoding
f.Write content
f.Close
End Sub
```

Sub SetConfig(in_path_work As String)

```
'sets the global variable values if they do not exist


If MC_Workfolder = "" Then
    MC_Workfolder = in_path_work
End If
'confirm the folder exists
If confirmFolder(MC_Workfolder) = False Then
    Global_Fail
    Exit Sub
End If
'confirm the log folder exists


If MC_LogName = "" Or MC_Log_folder = "" Then
    'set file based on day
    MC_Log_folder = MC_Workfolder & "\Logs"
    If confirmFolder(MC_Log_folder) = False Then Exit Sub
    MC_LogName = MC_Log_folder & "\" & Format(Date, "YYYYMMDD") & "_Log.txt"
    'set file to record unknown emails - for future actions
    MC_LogNewName = MC_Log_folder & "\" & Format(Date, "YYYYMMDD") & "_UnknownLog.txt"
    If createLogFile(MC_LogName) = False Then Exit Sub
    MC_DataName = MC_Log_folder & "\" & "Data.txt"
    If createLogFile(MC_DataName) = False Then Exit Sub
End If
```

Function Remove_non_ascii(in_str As String) As String

```
'This is to remove non ascii characters that will cause the log function to fail if they are not removed
Dim RegexVar As Object
```

```
Set RegexVar = CreateObject("VBScript.RegExp")

RegexVar.Pattern = "[^\x00-\x7F]+"

RegexVar.Global = True

RegexVar.IgnoreCase = IgnoreCase


result = RegexVar.Replace(in_str, "")

'Debug.Print stringVar & " is now " & result


Remove_non_ascii = result

End Function
```

Sub Global_Fail()

```
'not able to continue, config values must be fixed

Dim response As VbMsgBoxResult

response = MsgBox("Global variables cant be set please check", vbCritical, "Critical issue")

End Sub
```

The main Module

requires to start with the declaration of the Public variables

```
Public ACTIONS()

Public MC_Workfolder As String

Public MC_Log_folder As String

Public MC_LogName As String

Public MC_LogNewName As String

Public MC_DataName As String

Public MC_mail_action_config As String
```

Sub Director()

```
'**** Integorates the mail box for sender information ***

'********** D. Thierry 1/10/2020 **********
```

```
'*****************************************************

'Main program to run

Dim Run_limit As Long

Dim Mailbox As String, Workfolder As String


'Set location where logs and outputs will go

Workfolder = "C:\Dev\MailReader"

Mailbox = "africanmeats@gmail.com\inbox"

Run_limit = -1   'set to -1 to run all, any other positive int to stop after that count


Call Read_Mail(Workfolder, Mailbox, Run_limit)

End Sub

Sub Read_Mail(Workfolder As String, Mailbox As String, Run_limit As Long)

' Objective to extract sender, date and read data, write into file


Dim ToDelete As Boolean, Apply_limit As Boolean

Dim actionOf As String, message As String, Mode As String, operation As String

Dim choice As VbMsgBoxResult

Dim folderItemsCount As Long

Dim numRecipients As Integer, RecipientNo As Integer, i As Integer, stop_counter As Integer, dataset As
String

Dim thisEmail As Outlook.MailItem, folderItems As Outlook.Items


Call SetConfig(Workfolder)

Dim msOutlook As Outlook.NameSpace 'Establish Outlook NameSpace

Dim Folder As Outlook.MAPIFolder 'Establish Folder as a MAPIFolder


Set msOutlook = Application.GetNamespace("MAPI")
```

```
'config what to do

operation = "make list"


Set folderItems = MC_GetFolderPath(Mailbox).Items

folderItemsCount = folderItems.Count

Call AddLog(MC_LogName, "items in folder = " & folderItemsCount, "Log")


If Run_limit < 0 Then Apply_limit = False Else Apply_limit = True

stop_counter = 0

For i = folderItemsCount To 1 Step -1

   If folderItems.Item(i).Class = 43 Then

      Set thisEmail = folderItems.Item(i)

      If operation = "make list" Then

         dataset = thisEmail.Subject & "|" & thisEmail.ReceivedTime & "|" & thisEmail.sender & "|" &
thisEmail.SenderEmailAddress

         Call AddLog(MC_LogName, dataset, "")

      Else

         Debug.Print "no action set"

      End If

   End If


   If Apply_limit And stop_counter > Run_limit Then Exit For

   stop_counter = stop_counter + 1

Next i

Debug.Print "stop after " & stop_counter

Call AddLog(MC_LogName, "End Main processing loop ", "Log")

End Sub
```