Environmental Compliance: Acquisition, Storage, and Analysis of Waste Oil Data

Exchange 2007 Mailbox Setup

This guide is intended to be used in conjunction with the data transmission system and database developed by WPI students for the storage and analysis of shipboard waste data.

Developed For:
United States Coast Guard
Office of Vessel Activities (COMDT CG-543), in conjunction with the
Office of Investigations and Casualty Analysis (COMDT CG-545)

Prepared & Edited by:
Patrick Brodeur
Renée Lanza
Elizabeth Morris
Edward Osowski

Table of Contents

Table of Figures

1 Introduction

This guide is intended to be used in conjunction with the data transmission system and ECP Database developed by WPI students for the storage and analysis of shipboard waste data. It offers multiple ways of implementing the Outlook/Exchange integration component of the data transmission system, including a test setup using a personal mailbox, a semi-automatic system using a public folder and a macro, and an automated system using a public folder and script running on the Exchange server.

Within this document, some technical terminology is used and there are references to a data transmission system and the "ECP Database". As briefly stated above, this document is meant to be used with the system developed by WPI students in the Fall of 2011 for the acquisition, storage, and analysis of SWOMS (Special Waste Oil Monitoring System) and sounding log data from vessels on environmental compliance plans (ECP's). The specific elements in this user manual refer to the Outlook/Exchange components of the standard data submission plan included in the system. Please refer to the full report, section 5.3 New Standard Transmission Method for more details.

Some helpful terms:

- swomsparser – program developed as part of the project for parsing data out of SWOMS reports
- Outlook – the Microsoft Office program for email and other organization tasks
- Exchange – the Microsoft server software that handles email, calendars, and various other communication information
- Visual Basic for Applications (VBA) – scripting language used by Microsoft Office to programmatically perform many tasks; used here to perform actions on emails stored on a Microsoft Exchange server

2 Outlook Test Setup

In order to test the SWOMS email submission system, a personal mailbox was set up using an Outlook rule and VBA script. This setup may be used to demonstrate or test the behavior of the data transmission system from receiving a SWOMS email to the data's storage as spreadsheets. The similarity between this setup and that on the server lends itself to allow testing on a local machine without making experimental modifications to the script on the server.

2.1 Creating a File Structure

Before any Outlook/Exchange modifications are made, the working directories should be created. The code modules use a file tree similar to the one shown below in Figure 1. In this screenshot, the folder is on the 'M' drive of the computer. The way the parsing program is written, it will always create and modify the SWOMS_DATA spreadsheet in the same folder as itself but will create vessel-specific tables in the same folder as the input (source) data.

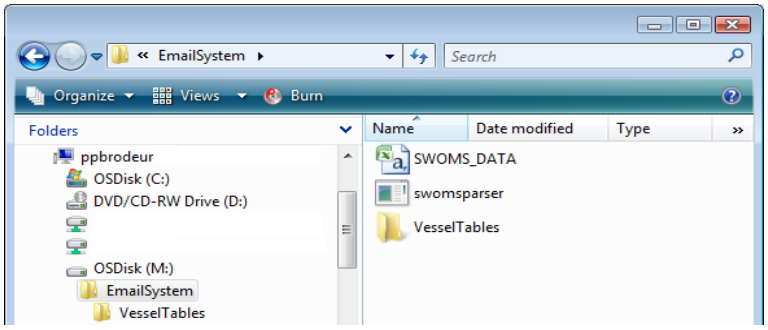

Figure 1: File Structure Example

The VBA script included uses the 'U' drive, which by default maps to a user's personal file share. The default structure in the included script uses "U:\EmailSystem\" as its primary working directory and "VesselTables\" as its vessel table subdirectory. In this setup, vessel-specific tables will be placed into the "VesselTables" directory, and the swomsparser program and SWOMS_DATA table will remain in the "EmailSystem" folder. More on the VBA script options with regard to folder structure may be found later in this document.

2.2 Creating the Outlook Rule

The next stage is to create Outlook rules to process emails when they arrive. Before making the rule, two tasks must be accomplished. The first of these pre-rule tasks is to create a folder to store the sample SWOMS emails. This folder can be named anything memorable to the user. The second pre-rule task is to create a placeholder script as described below in 2.2.1 Create Script Placeholder.

2.2.1 Create Script Placeholder

To create and add the script placeholder, open the Visual Basic for Applications editor. This can be done by pressing the keyboard shortcut Alt-F11 or by going to Tools > Macro > Visual Basic Editor in the top menu.

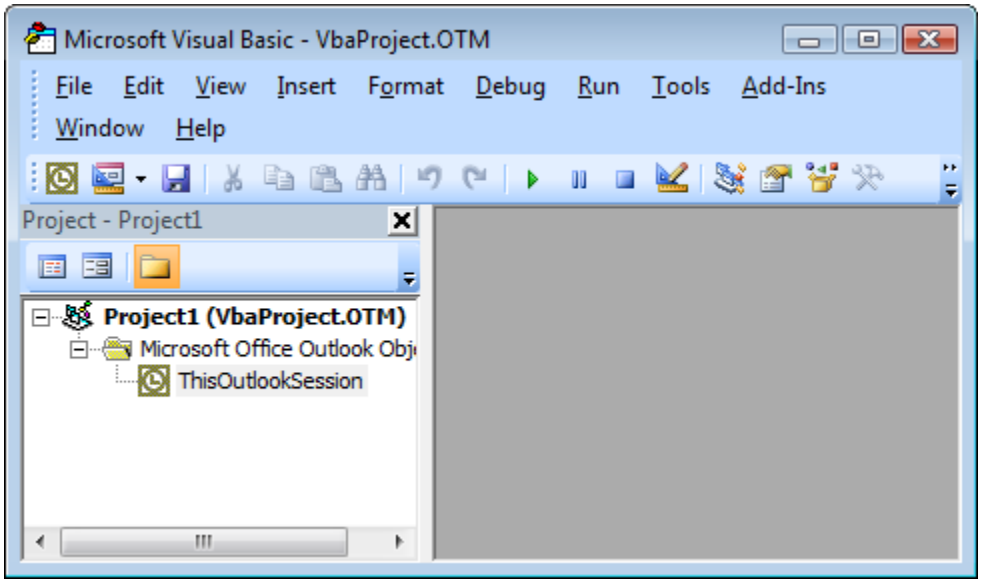

Figure 2: Opened VBA Editor

With the VBA editor opened, the "ThisOutlookSession" code module can then be opened. On the left-hand side, expand Project1 and its subfolder. "ThisOutlookSession" should appear in this list, as shown above in Figure 2. Double click to open it, and the following screen should appear:

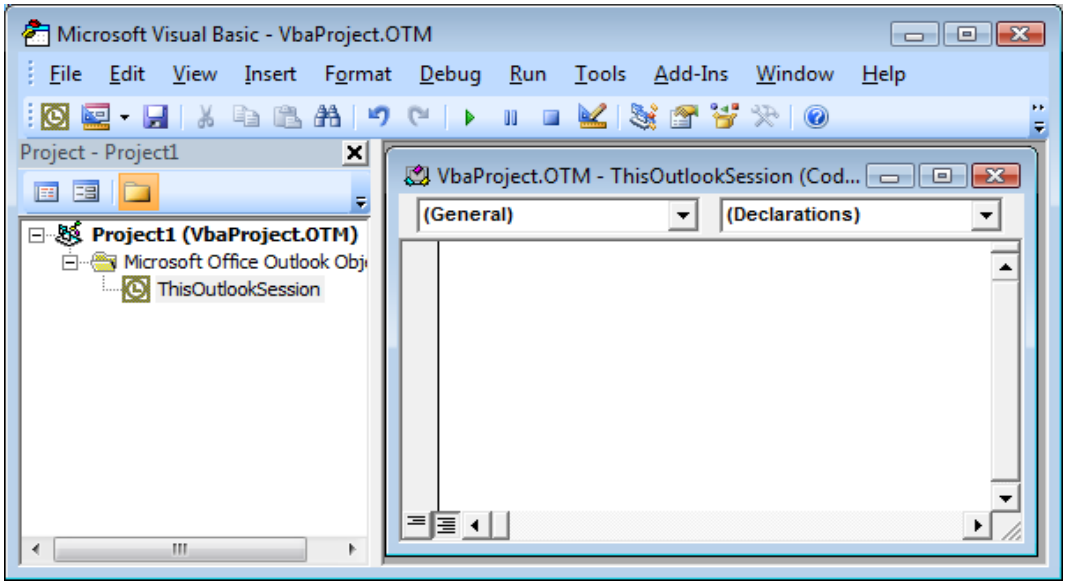

Figure 3: VBA Editor with ThisOutlookSession Opened

This code module should be empty by default. If there is any code already in place, verify with a system administrator that it is safe and belongs there. Any code with an unknown source is a significant security risk.

In the VBA editor's menu, click on Insert > Procedure. It should open up a simple dialog window. Fill in the name, select "Sub" for the type parameter, and select "Public" for the scope parameter.

The Add Procedure box will add in the start and end lines of the SWOMS rule. Modify the code to match the lines shown below in SWOMS Rule Code Segment 1 and save the script. The placeholder will now be a valid script to be called by a rule. The VBA editor window may be left open for later or closed.

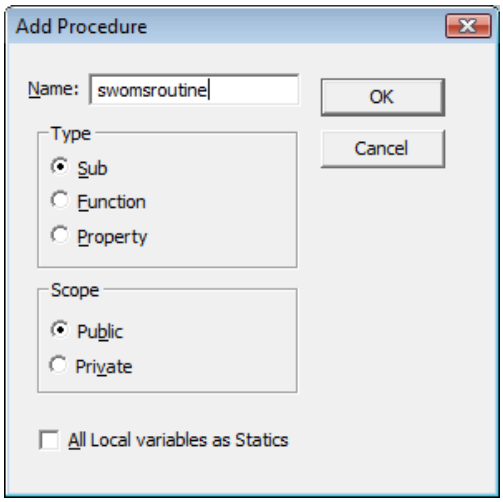

Figure 4: Add Procedure Dialog

```
Public Sub SWOMSRule(currentMail As Outlook.MailItem)
    Debug.Print "Script Run Successfully"
End Sub
```

Figure 5: SWOMS Rule Code Segment 1

2.2.2 Making the Rule

To begin making the rule, go to Tools > Rules & Alerts from the menu bar. Click "New Rule…" to begin, and follow the steps below Figure 6.

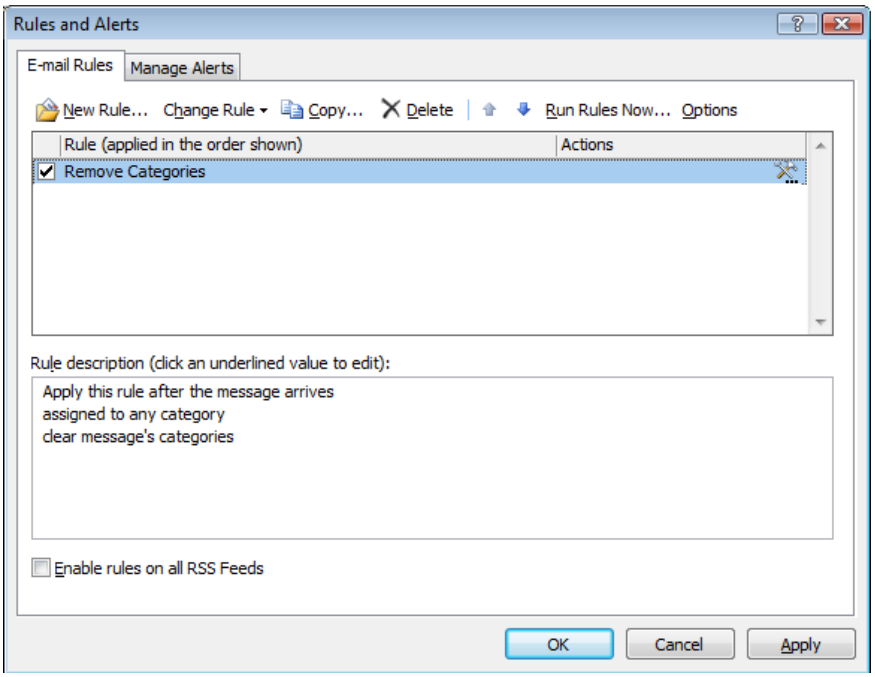

Figure 6: Rules and Alerts Main Screen

1. Select "Check messages when they arrive" under the "Start from a blank rule" heading and click "Next".
2. No conditions must be set, click "Next".
    a. Outlook may ask for confirmation; yes, this rule should be run on every email that is received.
3. Check the "run a script" box.
4. Click on "a script" and select the VBA script created earlier.
5. Click "Next".
6. The script checks if emails are marked as complete so no further exclusions must be made using a rule, so the exclusions page can be skipped.
7. On the final page, give the rule a descriptive name and keep "Turn on this rule" checked.
8. Click "Finish" to be returned to the Rules and Alerts dialog.

At this point, Outlook will pop up a warning message saying, "This rule is a client-only rule, and will process only when Outlook is running." As the warning indicates, this setup will only work when a user has Outlook running. For testing purposes, this is not an issue.

2.3 Adding Visual Basic Script

Re-open the Visual Basic for Applications editor. This can be done by pressing the keyboard shortcut Alt-F11 or by going to Tools > Macro > Visual Basic Editor in the top menu bar. At this point, two options are possible. The first option is to replace the previously-added code in ThisOutlookSession with the code in the accompanying file "ThisOutlookSession.txt". For a better understanding of what is happening in the script, or if the accompanying file is missing, continue reading this section. Modifications may be necessary before the code works properly, depending on the computer's specific setup.

2.3.1 Adding Code

In ThisOutlookSession, remove the "Debug.Print" line added earlier; it isn't necessary. Following the first line ("Public Sub…"), add the lines shown below in SWOMS Rule Code Segment 2. This section declares variables and sets the working directory (EmailSystem in the earlier example) and the vessel table subdirectory.

```
Dim vFile, vFile2, workingDir, vslTblSubdir As String
Dim i As Integer

workingDir = "U:\EmailSystem\"
vslTblSubdir = "VesselTables\"
```

Figure 7: SWOMS Rule Code Segment 2

SWOMS Rule Code Segment 3 below shows the "If" conditional that tests the email for the keywords that it is a SWOMS email, and also checks to make sure there isn't a "Completed" flag. The three lines following the "if" statement set the mail item to be plaintext and mark it as read.

```
If (  InStr(currentMail.Subject, "Daily Report") <> 0) And _
      currentMail.FlagStatus <> olFlagComplete Then

      currentMail.BodyFormat = olFormatPlain
      currentMail.UnRead = False
      currentMail.Save
```

Figure 8: SWOMS Rule Code Segment 3

SWOMS Rule Code Segment 4 creates a temporary text file with a randomly-generated number as part of its name and copies the body of the email into the text file. This text file will be processed by the swomsparser program.

```
      ' Random name for temporary text file
      vFile = vslTblSubdir & "TMP" & (Rnd * 1000) & ".txt"
      ' Put body of the email into a temporary text file
      Open (workingDir & vFile) For Output As #1
      Print #1, currentMail.Body
      Close #1
```

Figure 9: SWOMS Rule Code Segment 4

The next code segment creates a batch file that uses the Windows command line interpreter to run the swomsparser program and clean up the text file and itself. This section is lengthy, but comments (shown in green) explain what the code is doing.

```
	' Create batch file to process data and clean up intermediate files
	' Random name for temporary batch file
	vFile2 = workingDir & "TMP" & (Rnd * 1000) & ".bat"
	Open vFile2 For Output As #2
	' Go to the directory for the SWOMS parser

	If (Left(workingDir, 2) = "\\") Then
		' If it's a network directory then make a pushd drive map
		Print #2, "pushd " & workingDir
	Else
		' Otherwise use the directory as-is
		Print #2, Left(workingDir, 2)
			Print #2, "cd " & workingDir
	End If
	' Run the parser & wait for completion
	Print #2, "start /wait swomsparser " & ".\" & vFile
	' Delete intermediates
	Print #2, "del .\" & vFile	' Instruction to delete input file
	Print #2, "del %0"		' Instruction to delete self
	Close #2
```

Figure 10: SWOMS Rule Code Segment 5

The last section of code before the "End Sub" statement is seen below in Code Segment 6. The line beginning with "Shell" runs the batch file, and the rest of the code sets the email to "Complete". The script is complete at this point, and the rule can be run.

```
	' Run the batch script outlined above
	Shell (vFile2)

	currentMail.FlagStatus = olFlagComplete
	currentMail.Save
End If
```

Figure 11: SWOMS Rule Code Segment 6

3 Implementation on Public Folder

The ideal solution to the email integration system is to have a shared Outlook folder (a public folder) that automatically runs the parsing program on emails as they come in. This functionality was not able to be completed, but a script was written to be implemented when it is possible. See section 0 for details.

3.1 Creating Public Folder

A public folder for the ECP Database has already been created at:
\HQS\HQS-PF-fldr-CG-5\HQS-PF-fldr-CG-543\CGECP
Do not create another public folder unless necessary.

Before anything else can be set up, a public folder must be created. This can be done by going to File > New > Folder. In the Create New Folder dialog, fill in the folder name, leave it set to "Mail and Post Items", and then select where to place the folder.

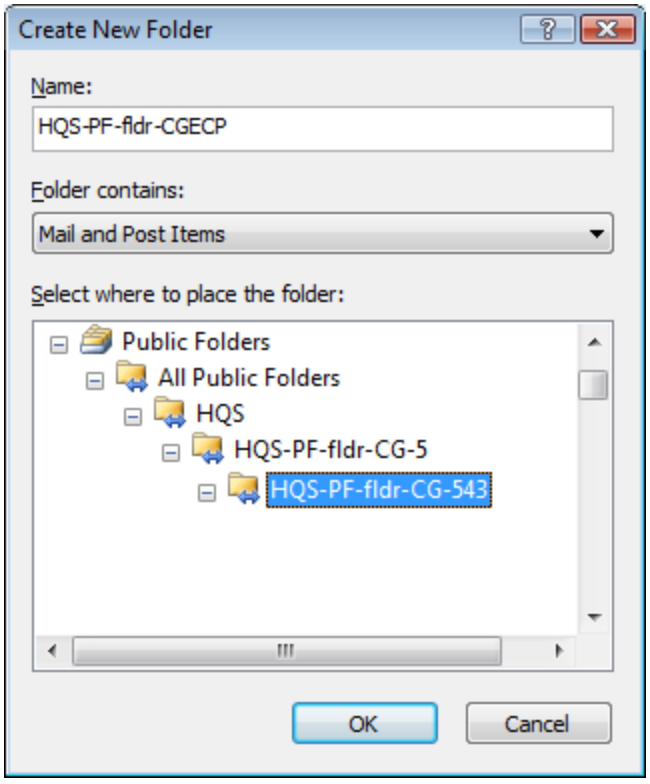

Figure 12: Add Folder Dialog

Once the public folder is created, there should be separate subfolders created to hold processed SWOMS emails and document (sounding log, ORB) emails. Within these folders, it may be desirable to create subfolders on a per-ship basis, though this is probably

unnecessary. The main folder will serve as triage and error control, where emails will enter and be directed and/or processed either automatically or by user intervention.

3.2 Outlook Macro (Semi-Automated)

Partial automation is achieved through the use of an Outlook Macro. First, the macro must be added into Outlook's VBA project. Sections 3.2.1 and 3.2.2 are two different options for putting the macro's code into the VBA project, and section 3.2.3 describes how to practically use the macro.

3.2.1 By Importing VBA Module

Importing the VBA macro as a VBA module is the easiest way to get it up and running immediately. Currently, the macro is set to use a shared folder on one of the Coast Guard servers. Modifications may be made to adjust for differences in how the file system is set up. At the time of this writing, no modification should be necessary for this macro to run "out of the box".

Open the Visual Basic for Applications editor. This can be done by pressing the keyboard shortcut Alt-F11 or by going to Tools > Macro > Visual Basic Editor in the top menu bar. Right click on "Project1" on the left-hand side and choose "Import File". Select the accompanying module file, "Mod_SWOMSMacro.bas". Close the VBA editor by pressing Alt-F11, Alt-F4, or by clicking the red exit button in the upper right hand corner. Continue to section 3.2.3.

3.2.2 Adding Code

Open the VBA editor. Go to Insert > Module to add a new VBA script module to the VBA project. Next, add SWOMS Macro Code Segment 1. This sets up the name of the script and includes variable definitions. Except for SWOMS Macro Code Segment 1 and SWOMS Macro Code Segment 10, indentation will be shifted one tab to the left for readability.

```
Public Sub SWOMSMacro()
      Dim vFile, vFile2, workingDir, vslTblSubdir As String
      Dim currentMail As MailItem
      Dim currentItem, currentSelection As Object
      Dim i As Integer
      Dim answer As VbMsgBoxResult
```

Figure 13: SWOMS Macro Code Segment 1

SWOMS Macro Code Segment 2 sets up the directories, as shown before for the SWOMS Rule. This time the working directory is pointing to a network folder.

```
workingDir = "\\Hqs-nas-t-001\cg-5\CG-54\CG-543\CG-5432\" _
      & "ECP - Environmental Compliance Program Master File\EmailSystem\"
vslTblSubdir = "VesselTables\"
```

Figure 14: SWOMS Macro Code Segment 2

The next code segment is unique to the macro. SWOMS Macro Code Segment 3 makes a list of all emails that are currently selected. It then checks the number, and advises the user that performing large sets of emails may be very slow. The script asks the user to confirm that they want to continue, and will then either quit or continue.

```
Set currentSelection = Application.ActiveExplorer.Selection
If (currentSelection.Count > 100) Then
      answer = MsgBox("Running this macro on a large number of emails (" _
            & currentSelection.Count & ") may take a long time and could" _
            & "cause Outlook to become unstable. " & vbCrLf & vbCrLf _
            & "Are you sure you want to continue?", vbYesNo, "Warning")
      If answer = vbNo Then Exit Sub
End If
```

Figure 15: SWOMS Macro Code Segment 3

The next section of code creates the first part of the batch file. The VBA creates the batch file in parts rather than all at once..

```
' Establish batch file
' Random name for temporary batch file
vFile2 = workingDir & "TMP" & (Rnd * 1000) & ".bat"
Open vFile2 For Output As #2
' Go to the directory for the SWOMS parser
If (Left(workingDir, 2) = "\\") Then
      Print #2, "pushd " & workingDir
Else
    Print #2, Left(workingDir, 2)
    Print #2, "cd " & workingDir
End If
```

Figure 16: SWOMS Macro Code Segment 4

Once the batch file is established, the macro begins going through what is selected. The code below in SWOMS Macro Code Segment 5 tells the script to only run the rest of it if the selected object is an email. “For Each” means to go through every selected email.

```
i = 0
For Each currentItem In currentSelection
    If currentItem.Class = olMail Then
```

Figure 17: SWOMS Macro Code Segment 5

Again, indentation is being shifted for readability. The next section warns the user if the program hits a 225-email limit. This is imposed for stability and to avoid hitting a server-side restrictions that limits emails processed to 249.

```
' Server restricts max emails processed in this way to 249 per call
i = i + 1
If i >= 225 Then
      MsgBox ("This macro may only be run on 225 messages at once." & _
            vbCrLf & "Not all selected emails were processed.")
      GoTo closeandfinish
End If
```

Figure 18: SWOMS Macro Code Segment 6

Similarly to in the SWOMS RULE, SWOMS Macro Code Segment 7 checks that the email is a SWOMS email and sets the same settings (plaintext, marked as read).

```
Set currentMail = currentItem
If ((InStr(currentMail.Subject, "Daily Report") <> 0) And _
            currentMail.FlagStatus <> olFlagComplete) Then
      currentMail.BodyFormat = olFormatPlain
      currentMail.UnRead = False
      currentMail.Save
```

Figure 19: SWOMS Macro Code Segment 7

The macro then goes through the same process as before to produce a text file containing the body of the email.

```
      vFile = "VesselTables\" & "TMP" & (Rnd * 1000) & ".txt" ' Random name
      for temporary text file
      ' Put body of the email into a temporary text file
      Open (workingDir & vFile) For Output As #1
      Print #1, currentMail.Body
      Close #1
```

Figure 20: SWOMS Macro Code Segment 8

As mentioned previously, the macro adds onto the batch file. SWOMS Macro Code Segment 9 shows the addition of two lines: one to start swomsparser, and the other to delete the temporary text file.

```
' Add to batch file to process data and clean up intermediate files
' Run the parser on the email, wait until it finishes
Print #2, "start /wait swomsparser " & ".\" & vFile
Print #2, "del .\" & vFile    ' Instruction to delete input file
```

Figure 21: SWOMS Macro Code Segment 9

At this point the email needs to be marked completed. SWOMS Macro Code Segment 10 returns to one-left-shifted indentation and shows the end of the two “if” statements and the “Next” statement associated with the “For Each” loop.

```
                currentMail.FlagStatus = olFlagComplete
                currentMail.Close (olSave)
                End If
            End If
        Next
```

Figure 23: SWOMS Macro Code Segment 10

The last section completes the batch file and runs it, ending the macro. The batch file will go through every email's text dump running the swomsparser on it.

```
closeandfinish:
      Print #2, "del %0"
      Close #2
      Shell (vFile2) ' Run the batch script outlined above
End Sub
```

Figure 22: SWOMS Macro Code Segment 11

3.2.3 Using Macro

Using the macro is as easy as selecting emails and running the macro. The macro can be run by selecting emails, then going to Tools > Macros and selecting the SWOMSMacro.

To make it easier, a quick button can be added to the toolbar. Right click in the "whitespace" on any part of the toolbar (see Figure 24).

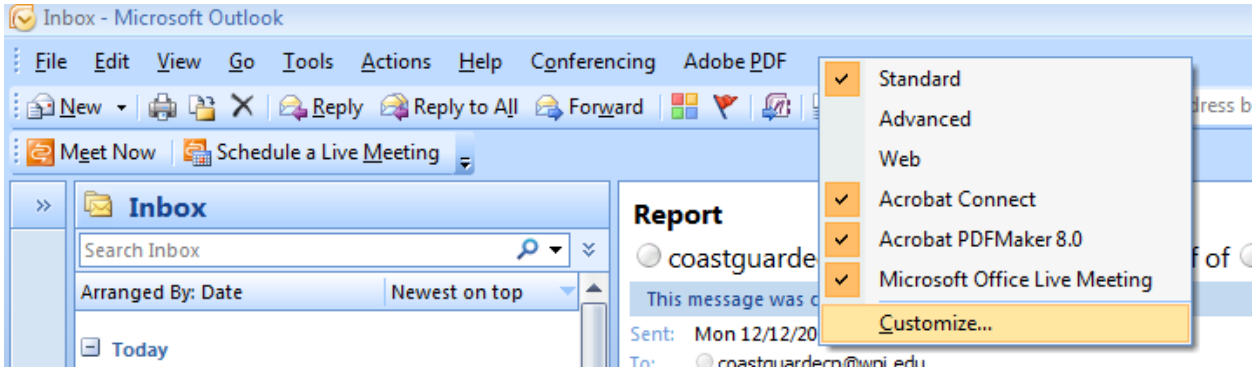

Figure 24: Menu to Customize Toolbar

Select "Customize…" and a dialog will open. Switch to the "Commands" tab, and then select "Macros" in the menu on the left. The window should look as it does in Figure 25.

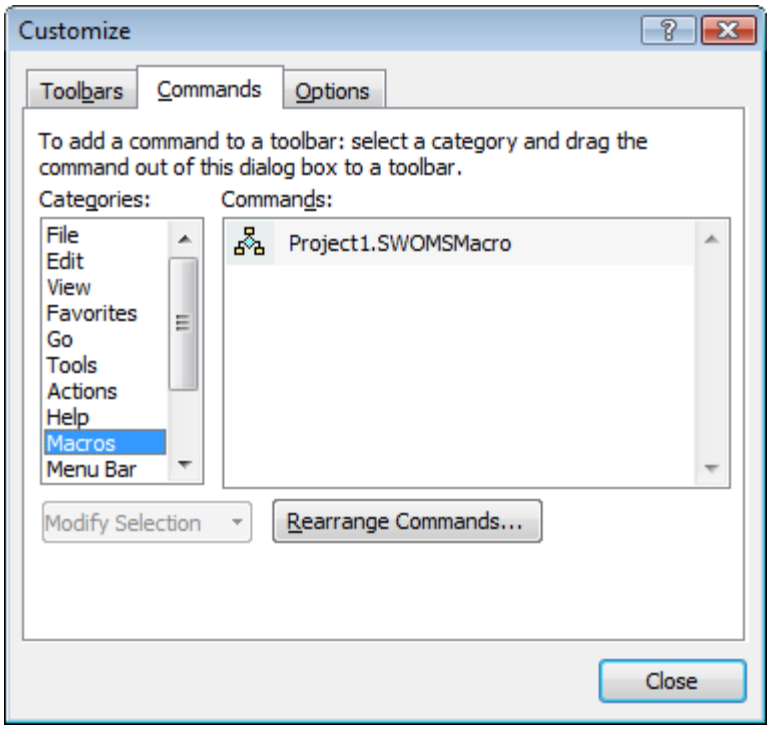

Figure 25: Add Macro to Toolbar

To add the SWOMSMacro to the toolbar, simply click the SWOMSMacro from the commands window and drag it to a toolbar. There should now be a button in the toolbar to run the macro. Figure 26 shows the macro after it has been dragged into the menu bar.

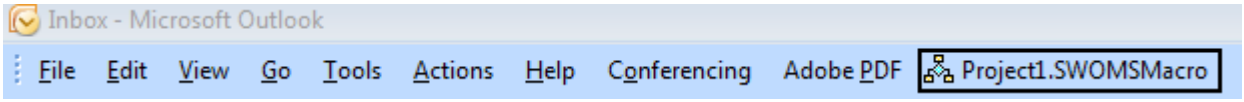

Figure 26: Macro Button in Toolbar

If at any point, the user would like to make any changes to the Macro button (including the name) or would like to remove the button, simply click the "Rearrange Commands" button in the Customization window that was originally used to add the Macro to the toolbar (see Figure 25). Select the toolbar that the Macro has been dragged onto under the Toolbars option (in this instance, it would be located under the Menu Bar), then select the Macro in the list to the left of the window, and then click on "Modify Selection" (see Figure 27).

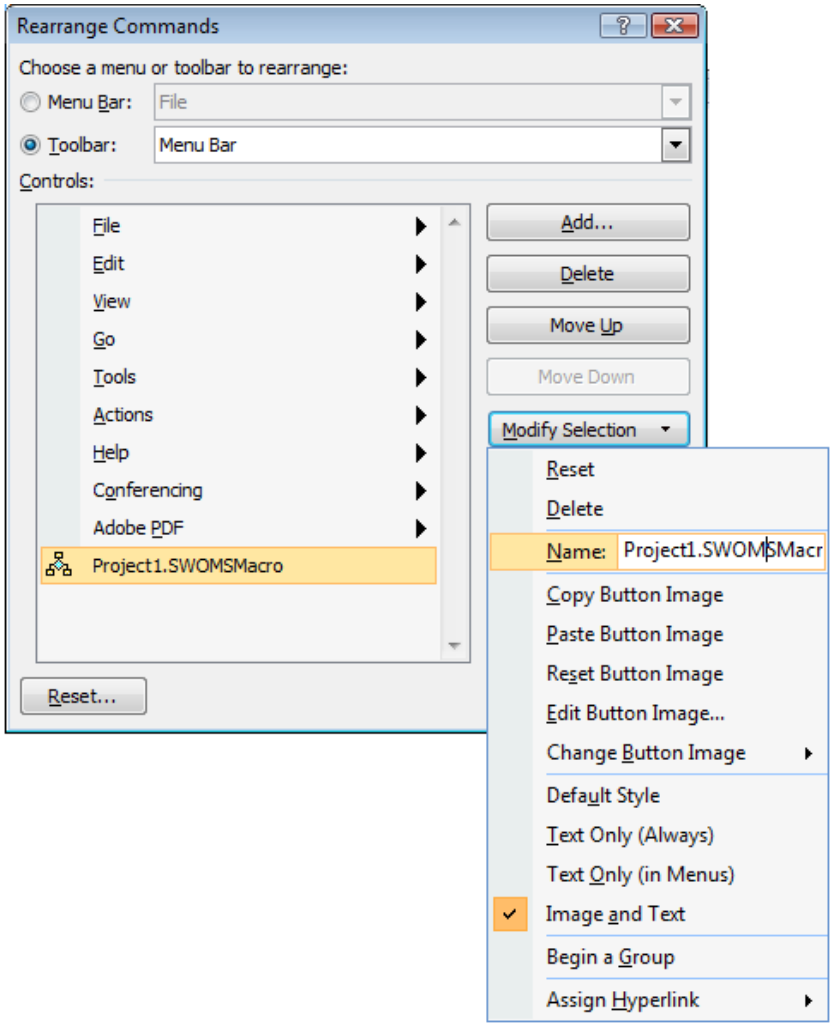

Figure 27: Modify Selection Menu

Once done, close the Customization window. At this point, simply select emails and click the button in order to automatically run the macro.

Note that the macro should not be used while a folder script is active, or else data collisions could occur, causing errors. Also included in the Mod_SWOMSMacro.bas file is a function and macro for removing "lock" files created by the swomsparser program. Normally, swomsparser will automatically remove the lock files that it creates. If program execution is halted before that point then the locks will still be present, preventing any future instances of the program from running and using the files. The ManualUnlock macro can help to remove the lock files.

Using the macro is as easy as selecting emails and running the macro. The macro can be run by selecting the emails, then going to Tools > Macros and selecting the SWOMSMacro. To make it easier, a quick button can be added to the toolbar. Right click in the "whitespace" on any part of the toolbar. Select "Customize Toolbar" and a dialog will open. On the left-hand side, find "Macros" in the dropdown. Select SWOMSMacro and click

the “Add” button. There should now be a button in the toolbar to run the macro. Simply select emails, and click the button and it should work automatically.

Note that the macro should not be used while a folder script is active, or else data collisions could occur, causing errors. Also included in the Mod_SWOMSMacro.bas file is a function and macro for removing “lock” files created by the swomsparser program. Normally, swomsparser will automatically remove the lock files that it creates. If program execution is halted before that point then the locks will still be left, preventing any future instances of the program from running and using the files.

1.1.1 Exchange Script (Automated)

Setting up the exchange script for full automation is the same process as it is done for the **Error! Reference source not found.**, but must be done on a service/shared account. Integration using public folders is an older communication service that is being phased out by Microsoft and is limited in its capabilities. The public folder administration tools, including the folder assistant, do not provide any way to run a script. In addition, running a VBA script to do any significant processing tasks is considered too much of a drain on a typical Exchange server to be practical from a stability standpoint.

The solution to this issue is to run an Outlook client on either a dedicated physical server or on a virtualized server. The Outlook client would be logged into a shared or “service” account on the Exchanger server and would be able to run the script in the same way as it would run for a personal user’s setup. The mailbox associated with the service account could still be set up to be viewable and changeable by personnel who use the system. Administration could be done by remote desktop sessions into the dedicated server, and would allow full control over changes to the VBA scripting and rules within the client.

3.3