
Scientia

May 02, 2021

Contents

1	Introduction	1
2	Outlook VBA Scripts	3
2.1	Introduction	3
2.2	Source	3
3	VIM Scripts	15
3.1	Introduction	15
3.2	GVim RC	15
3.3	GVim Personal Setup	15
4	Metaquotes MT4 and MT5	17
4.1	Introduction	17
4.2	MT4 Setup	17
	Index	19

CHAPTER 1

Introduction

This is a work in progress

Hello World

2.1 Introduction

This VBA Module provides a feature to write a selection of emails to disk, such that the .msg file is named with date_sent – sender – subject and the visible attachments are saved alongside the file with the same date_sent as a prefix.

The .msg file and the associated attachments have their modified timestamp set to the sent timestamp.

The email in the Outlook Mailbox has the category “saved” added to it, so that messages which have already been saved can be kept track of.

Repository on GitHub: [outlook-vba](https://github.com/simon-hoffe/outlook-vba).

With thanks to Chip Pearson, www.cpearson.com, chip@cpearson.com for his file timestamp get/set VBA functions.

2.2 Source

```

1 Attribute VB_Name = "modEmailExport"
2 Option Explicit
3
4 ' modEmailExport
5 ' By Simon Hoffe, https://github.com/simon-hoffe
6
7
8
9 '-----
10 ''' SaveSelectedEmails()
11 '''
12 ''' Call this sub when one or more emails are selected in the navigator.
13 ''' Each email will be saved to the folder "\Documents\Emails\" in the user's
14 ''' home directory.
15 '''

```

(continues on next page)

(continued from previous page)

```
16 ''' Each email will be saved to a file of the form:
17 '''     E yyyyymmdd-hhmm - INITIAL@domain - Subject
18 '''     where
19 '''         yyyyymmdd-hhmm is the time of sending, in the timezone of the user
20 '''         INITIAL@domain is a the Initials @ the domain of the sender
21 '''         Subject is the subject of the email. Repeating "RE" and "FWD" prefixes are
↳reduced to one.
22 '''
23 '''     Then all visible attachments, and/or all non image attachments will also be
↳saved
24 '''     to the \Documents\Emails\ folder in the form:
25 '''
26 '''         E yyyyymmdd-hhmm ___ filename.ext
27 '''         where
28 '''             yyyyymmdd-hhmm is the same timestamp from the email.
29 '''             filename.ext is the original filename of the attachment.
30 '''
31 '''     There is some rudimentary checking for the existance of the folder, and
32 '''     for existing files with the same name. There is the option to overwrite the
↳original .msg files.
33 '''     Attachments never overwrite and will be issued with unique names by appending
↳a number "-1" or "-2" and so on.
34
35
36 Public Sub SaveSelectedEmails()
37     Dim oMail As Outlook.MailItem
38     Dim objItem As Object
39     Dim enviro As String
40     Dim sPath As String
41     Dim nMsgResult As Long
42
43     Dim oRegEx As Object
44     Set oRegEx = CreateObject("vbscript.regexp")
45
46
47     enviro = CStr(Environ("USERPROFILE"))
48     sPath = enviro & "\Documents\Emails\"
49
50     ' Prompt if the directory doesn't exist.
51     If Not TestDirExist(sPath) Then
52         nMsgResult = MsgBox( _
53             Prompt:=sPath & " does not exist.", _
54             Title:"Critical Error", _
55             Buttons:=vbCritical + vbOKOnly)
56         Exit Sub
57     End If
58
59     ' Iterate through all the selected emails
60     For Each objItem In ActiveExplorer.Selection
61         If objItem.MessageClass = "IPM.Note" Then
62             Set oMail = objItem
63
64             SaveOneMessageAsMsg oMail
65         End If
66     Next
67 End Sub
68
```

(continues on next page)

(continued from previous page)

```

69 '-----
70 ''' SaveOpenEmail()
71 '''
72 ''' Call this sub from a window with the one email open and display.
73 ''' Behaviour is as for SaveSelectedEmails() above.
74 '''
75 Public Sub SaveOpenEmail()
76     Dim oMail As Outlook.MailItem
77     Dim objItem As Object
78     Dim enviro As String
79     Dim sPath As String
80     Dim nMsgResult As Long
81
82     Dim oRegex As Object
83     Set oRegex = CreateObject("vbscript.regexp")
84
85
86     enviro = CStr(Environ("USERPROFILE"))
87     sPath = enviro & "\Documents\Emails\"
88
89     If Not TestDirExist(sPath) Then
90         nMsgResult = MsgBox( _
91             Prompt:=sPath & " does not exist.", _
92             Title:="Critical Error", _
93             Buttons:=vbCritical + vbOKOnly)
94         Exit Sub
95     End If
96
97     Set objItem = ActiveInspector.CurrentItem
98
99     If objItem.MessageClass = "IPM.Note" Then
100         Set oMail = objItem
101
102         SaveOneMessageAsMsg oMail
103     End If
104 End Sub
105
106 '-----
107 ''' ShowSenders()
108 '''
109 ''' Call from the navigator with one or more emails selected.
110 ''' This is a development tool to display the information pulled from the
111 ''' MailItem object for each of the emails.
112 ''' It doesn't write anything
113
114 Public Sub ShowSenders()
115     Dim oMail As Outlook.MailItem
116     Dim objItem As Object
117     Dim sPath As String
118     Dim dtDate As Date
119     Dim sName As String
120     Dim enviro As String
121     Dim sSender As String
122     Dim sSubject As String
123     Dim sDateTime As String
124
125     enviro = CStr(Environ("USERPROFILE"))

```

(continues on next page)

(continued from previous page)

```

126 For Each objItem In ActiveExplorer.Selection
127     If objItem.MessageClass = "IPM.Note" Then
128         Set oMail = objItem
129
130         GetDateString oMail, sDateTime
131         GetSenderString oMail, sSender
132         GetSubjectString oMail, sSubject
133
134         sName = "E " & sDateTime & " - " & sSender & " - " & sSubject & ".msg"
135
136         sPath = enviro & "\Documents\Emails"
137
138         MsgBox "Subject: " & oMail.Subject & vbCrLf & _
139             "SentOn: " & oMail.SentOn & vbCrLf & _
140             "SenderName: " & oMail.SenderName & vbCrLf & _
141             "SenderAddress: " & oMail.SenderEmailAddress & vbCrLf & _
142             "SenderEmailType: " & oMail.SenderEmailType & vbCrLf & _
143             vbCrLf & _
144             "Folder: " & sPath & vbCrLf & _
145             "Filename: " & sName
146
147     End If
148 Next
149
150 End Sub
151
152 '-----
153 ''' SaveOneMessageAsMsg(oMail As Outlook.MailItem
154 '''
155 ''' The workhorse called by SaveSelectedEmails() and SaveOpenEmail()
156
157 Private Sub SaveOneMessageAsMsg(ByRef oMail As Outlook.MailItem)
158     Dim oAttachment As Outlook.Attachment
159     Dim objItem As Object
160     Dim sPath As String
161     Dim dtDate As Date
162     Dim sName As String
163     Dim sAttName As String
164     Dim sAttExt As String
165     Dim sExt As String
166     Dim enviro As String
167     Dim sSender As String
168     Dim sSubject As String
169     Dim sDateTime As String
170     Dim bSkipAttachment As Boolean
171     Dim iSize As Long
172     Dim iSizeLimit As Long
173     Dim nMsgResult As Long
174     Dim bSkipEmail As Boolean
175     Dim vTemp As Variant
176     Dim vPropNames() As Variant
177     Dim bHiddenProp As Boolean
178     Dim Result As Boolean
179
180
181     Dim oRegex As Object
182     Set oRegex = CreateObject("vbscript.regex")

```

(continues on next page)

(continued from previous page)

```

183
184
185     enviro = CStr(Environ("USERPROFILE"))
186     sPath = enviro & "\Documents\Emails\"
187
188     If Not TestDirExist(sPath) Then
189         nMsgResult = MsgBox( _
190             Prompt:=sPath & " does not exist.", _
191             Title:="Critical Error", _
192             Buttons:=vbCritical + vbOKOnly)
193         Exit Sub
194     End If
195
196     GetDateString oMail, sDateTime
197     GetSenderString oMail, sSender
198     GetSubjectString oMail, sSubject
199
200     sName = "E " & sDateTime & " - " & sSender & " - " & sSubject
201
202
203     sExt = ".msg"
204
205     ' Some code for debugging
206
207     '         MsgBox "Subject: " & oMail.Subject & vbCrLf & _
208     '             "SentOn: " & oMail.SentOn & vbCrLf & _
209     '             "SenderName: " & oMail.SenderName & vbCrLf & _
210     '             "SenderAddress: " & oMail.SenderEmailAddress & vbCrLf & _
211     '             "SenderEmailType: " & oMail.SenderEmailType & vbCrLf & _
212     '             vbCrLf & _
213     '             "Folder: " & sPath & vbCrLf & _
214     '             "Filename: " & sName & sExt
215     '
216     '         Debug.Print sPath & sName
217
218     bSkipEmail = False
219     If TestFileExist(sPath & sName & sExt) Then
220         nMsgResult = MsgBox( _
221             Prompt:="This file already exists:" & vbCrLf & vbCrLf & _
222             sName & sExt & vbCrLf & vbCrLf & _
223             "Overwrite?", _
224             Title:="Warning: File Exists", _
225             Buttons:=vbQuestion + vbYesNoCancel)
226
227         If nMsgResult = vbCancel Then
228             Exit Sub
229         End If
230
231         If nMsgResult = vbYes Then
232             bSkipEmail = False
233         Else
234             bSkipEmail = True
235         End If
236     End If
237
238     If Not bSkipEmail Then
239         ' Save the email to file on disk

```

(continues on next page)

```

240 oMail.SaveAs sPath & sName & sExt, olMSG
241
242 ' Set the Modified Timestamp of the file to the Sent Timestamp
243 Result = SetFileDateTime(FileName:=sPath & sName & sExt, _
244     FileDateTime:=oMail.SentOn, _
245     WhichDateToChange:=FileDateLastModified, _
246     NoGMTConvert:=False)
247
248 If Result = False Then
249     Debug.Print "An error occurred with SetFileDateTime."
250 Else
251     ' If all is well, add a category called "Saved" to the email in the
252     ' Outlook Mailbox to flag that it's been saved
253     AddCategoryToEmail oMail, "Saved"
254     oMail.Save
255 End If
256
257 For Each oAttachment In oMail.Attachments
258     bSkipAttachment = False
259     oRegEx.Pattern = "^(*?) (\.?[^.]*)$"
260     sAttName = oRegEx.Replace(oAttachment.FileName, "$1")
261     sAttExt = LCase(oRegEx.Replace(oAttachment.FileName, "$2"))
262     iSize = oAttachment.Size
263
264
265     ' Differentiate between attachments which are embedded inline, and
↳explicit attachments
266     Const PR_ATTACHMENT_HIDDEN = "http://schemas.microsoft.com/mapi/proptag/
↳0x7FFE00B"
267
268     vPropNames = Array(PR_ATTACHMENT_HIDDEN)
269     vTemp = oAttachment.PropertyAccessor.GetProperties(vPropNames)
270
271     bHiddenProp = Not IsError(vTemp(0))
272
273     bSkipAttachment = False
274
275     If bHiddenProp Then
276         Select Case sAttExt
277             Case ""
278                 bSkipAttachment = True
279             Case "."
280                 bSkipAttachment = True
281             Case ".png"
282                 bSkipAttachment = True
283             Case ".jpg"
284                 bSkipAttachment = True
285             Case ".gif"
286                 bSkipAttachment = True
287             Case Else
288                 ' Don't skip hidden attachments which aren't in the above list
289                 bSkipAttachment = False
290         End Select
291     End If
292
293     ' Ignore files which look like: F4F90AC9@CEDD0C72.18bc1a5d
294     ' regardless of whether tagged as hidden or not

```

(continues on next page)

(continued from previous page)

```

295     oRegex.Pattern = "^\[0-9a-fA-F]{5,}$"
296     If oRegex.Test(sAttExt) Then
297         bSkipAttachment = True
298     End If
299
300     ' Ignore files which have no extension
301     ' regardless of whether tagged as hidden or not
302     oRegex.Pattern = "^\.{0,1}$"
303     If oRegex.Test(sAttExt) Then
304         bSkipAttachment = True
305     End If
306
307     If Not bSkipAttachment Then
308         sName = "E " & sDateTime & " ____ " & sAttName
309
310         MakeFileNameUnique sPath, sName, sAttExt
311
312         oAttachment.SaveAsFile sPath & sName & sAttExt
313
314         Result = SetFileDateTime(FileName:=sPath & sName & sAttExt, _
315             FileDateTime:=oMail.SentOn, _
316             WhichDateToChange:=FileDateLastModified, _
317             NoGMTConvert:=False)
318
319         If Result = False Then
320             Debug.Print "An error occurred with SetFileDateTime."
321         End If
322     End If
323 Next
324 End If
325 End Sub
326
327
328 '-----
329 Private Sub ReplaceCharsForFileName(sName As String, _
330     sChr As String _
331 )
332     sName = Replace(sName, "'", sChr)
333     sName = Replace(sName, "*", sChr)
334     sName = Replace(sName, "/", sChr)
335     sName = Replace(sName, "\", sChr)
336     sName = Replace(sName, ":", sChr)
337     sName = Replace(sName, "?", sChr)
338     sName = Replace(sName, Chr(34), sChr)
339     sName = Replace(sName, "<", sChr)
340     sName = Replace(sName, ">", sChr)
341     sName = Replace(sName, "|", sChr)
342 End Sub
343
344 '-----
345 Private Sub GetDateString(oMail As Outlook.MailItem, _
346     sDateString As String _
347 )
348     Dim dtDate As Date
349
350     dtDate = oMail.SentOn
351     sDateString = Format(dtDate, "yyyymmdd", vbUseSystemDayOfWeek, vbUseSystem) & _

```

(continues on next page)

```

352     Format(dtDate, "-hhnn", vbUseSystemDayOfWeek, vbUseSystem)
353 End Sub
354
355 '-----
356 Private Sub AddCategoryToEmail(ByRef oMail As Outlook.MailItem, _
357     sNewCategory As String _
358 )
359     Dim sCategorySeparator As String
360     Dim sCategories() As String
361     Dim bNewCatExist As Boolean
362     Dim sOne As Variant
363
364     sCategorySeparator = RegKeyRead("HKEY_CURRENT_USER\Control_
↵Panel\International\sList")
365     If Len(sCategorySeparator) <> 1 Then
366         sCategorySeparator = ","
367     End If
368
369     sCategories = Split(oMail.Categories, sCategorySeparator)
370
371     bNewCatExist = False
372     For Each sOne In sCategories
373         If StrComp(UCase(sOne), UCase(sNewCategory), vbTextCompare) = 0 Then
374             bNewCatExist = True
375             Exit For
376         End If
377     Next sOne
378
379     If Not bNewCatExist Then
380         ReDim Preserve sCategories(UBound(sCategories) + 1)
381         sCategories(UBound(sCategories)) = sNewCategory
382
383         oMail.Categories = Join(sCategories, sCategorySeparator)
384     End If
385 End Sub
386
387 '-----
388 Private Sub GetSenderString(oMail As Outlook.MailItem, _
389     sSenderString As String _
390 )
391     Dim objRegExp As Object
392     Dim oMatches As Object
393     Dim oMatch As Object
394     Dim sSenderName As String
395     Dim sInitials As String
396     Dim sSenderAddress As String
397     Dim sDomain As String
398     Dim sTemp As String
399
400     Set objRegExp = CreateObject("vbscript.regexp")
401
402     ' MsgBox "Subject: " & oMail.Subject & vbCrLf & _
403     '     "SentOn: " & oMail.SentOn & vbCrLf & _
404     '     "SenderName: " & oMail.SenderName & vbCrLf & _
405     '     "SenderAddress: " & oMail.SenderEmailAddress & vbCrLf & _
406     '     "SenderEmailType: " & oMail.SenderEmailType
407

```

(continues on next page)

(continued from previous page)

```

408     ' Process the Sender Name
409
410     With objRegex
411         .Global = True
412         .MultiLine = True
413         .IgnoreCase = True
414     End With
415
416     sSenderName = oMail.SenderName
417     sSenderAddress = oMail.SenderEmailAddress
418
419     ' Remove any email address add on in the sender name
420     objRegex.Pattern = "\S+@[^\.\]\S*\.\w{2,}"
421     If objRegex.Test(sSenderName) Then
422         sTemp = objRegex.Replace(sSenderName, "")
423     End If
424
425     ' Trim any leading and/or trailing white space
426     objRegex.Pattern = "^\s*(.*?)\s*$"
427     sTemp = objRegex.Replace(sTemp, "$1")
428
429     If Len(sTemp) = 0 Then
430         objRegex.Pattern = "(\\S+)@[^\.\]\S*\.\w{2,}"
431         sTemp = objRegex.Replace(sSenderName, "$1")
432     End If
433
434     sSenderName = sTemp
435
436     ' Remove anything between brackets
437     objRegex.Pattern = "(\\([^\(\)]*\)|\\[[^\[\]]*\]|\\{[^}]*\\}|<[^\<>]*>)"
438     If objRegex.Test(sSenderName) Then
439         sSenderName = objRegex.Replace(sSenderName, "")
440     End If
441
442     ' If the Sender name is in "Surname, Name" format, then switch it around
443     objRegex.Pattern = "^(^,)+, (^,)+$"
444     If objRegex.Test(sSenderName) Then
445         sSenderName = objRegex.Replace(sSenderName, "$2 $1")
446     End If
447
448     objRegex.Pattern = "\\b\\w"
449     Set oMatches = objRegex.Execute(sSenderName)
450
451     sInitials = ""
452     For Each oMatch In oMatches
453         sInitials = sInitials & UCase(oMatch.Value)
454     Next
455
456     sDomain = ""
457     If oMail.SenderEmailType = "SMTP" Then
458         objRegex.Pattern = "@[^\.]*"
459         Set oMatches = objRegex.Execute(sSenderAddress)
460         For Each oMatch In oMatches
461             sDomain = LCase(oMatch.Value)
462         Exit For
463     Next
464 Else

```

(continues on next page)

```
465     sDomain = "@local"
466     End If
467
468     sSenderString = sInitials & sDomain
469 End Sub
470
471 '-----
472 Private Sub GetSubjectString(oMail As Outlook.MailItem, _
473 sSubjectString As String _
474 )
475     Dim objRegex As Object
476     Dim sName As String
477
478     sName = oMail.Subject
479
480     Set objRegex = CreateObject("vbscript.regexp")
481
482     With objRegex
483         .Global = True
484         .MultiLine = True
485         .IgnoreCase = True
486     End With
487
488     ' Remove duplicate "RE" strings
489     objRegex.Pattern = "(re\W+) (re\W+)+"
490     sName = objRegex.Replace(sName, "$1")
491
492     ' Remove duplicate "FW" strings
493     objRegex.Pattern = "(fwd?\W+) (fwd?\W+)+"
494     sName = objRegex.Replace(sName, "$1")
495
496     ReplaceCharsForFileName sName, " "
497
498     ' Trim any leading and/or trailing white space
499     objRegex.Pattern = "^\s*(.*?)\s*$"
500     sName = objRegex.Replace(sName, "$1")
501
502     ' Trim any double spaces
503     objRegex.Pattern = "\s+"
504     sName = objRegex.Replace(sName, " ")
505
506     ' Limit the length to 100 characters
507     sName = Left(sName, 100)
508
509     If Len(sName) = 0 Then
510         sName = "No Subject"
511     End If
512
513     sSubjectString = sName
514 End Sub
515
516 '-----
517 Private Sub MakeFileNameUnique(sPath As String, sFileName As String, sExt As String _
518 )
519     Dim oRegex As Object
520     Dim sIntExt As String
521     Dim sIntPath As String
```

(continues on next page)

(continued from previous page)

```

522 Dim sIntFileName As String
523 Dim sTestName As String
524 Dim i As Integer
525
526 Set oRegex = CreateObject("vbscript.regexp")
527
528 If Len(sExt) <> 0 Then
529     oRegex.Pattern = "^\.*"
530     sIntExt = oRegex.Replace(sExt, ".") ' Make sure the ext has a "."
531 Else
532     sIntExt = ""
533 End If
534
535
536 oRegex.Pattern = "\\*$"
537 sIntPath = oRegex.Replace(sPath, "") ' Remove the trailing backslash, for now
538
539 If Len(Dir(sIntPath, vbDirectory)) = 0 Then
540     ' Bigger problems, the directory doesn't exist!
541     Exit Sub
542 End If
543
544 sIntPath = sIntPath + "\"
545 sIntFileName = sFileName
546 sTestName = Dir(sIntPath & sIntFileName & sIntExt)
547
548 i = 0
549 Do While sTestName <> ""
550     i = i + 1
551     sIntFileName = sFileName & "-" & i
552     sTestName = Dir(sIntPath & sIntFileName & sIntExt)
553 Loop
554 sFileName = sIntFileName
555 End Sub
556
557 '-----
558 Private Function TestDirExist(sPath As String) As Boolean
559     Dim sIntPath As String
560     Dim oRegex As Object
561     Set oRegex = CreateObject("vbscript.regexp")
562
563     oRegex.Pattern = "\\*$"
564     sIntPath = oRegex.Replace(sPath, "") ' Remove any trailing backslash
565     If Len(Dir(sIntPath, vbDirectory)) = 0 Then
566         TestDirExist = False
567     Else
568         TestDirExist = True
569     End If
570 End Function
571
572 '-----
573 Private Function TestFileExist(sPath As String) As Boolean
574     If Len(Dir(sPath)) = 0 Then
575         TestFileExist = False
576     Else
577         TestFileExist = True
578     End If

```

(continues on next page)

```
579 End Function
580
581
582 '-----
583 Function BrowseForFolder(Optional OpenAt As Variant) As Variant
584     Dim ShellApp As Object
585     Set ShellApp = CreateObject("Shell.Application"). _
586     BrowseForFolder(0, "Please choose a folder", 0, OpenAt)
587
588     On Error Resume Next
589     BrowseForFolder = ShellApp.self.Path
590     On Error GoTo 0
591
592     Set ShellApp = Nothing
593     Select Case Mid(BrowseForFolder, 2, 1)
594         Case Is = ":"
595             If Left(BrowseForFolder, 1) = ":" Then GoTo Invalid
596         Case Is = "\"
597             If Not Left(BrowseForFolder, 1) = "\" Then GoTo Invalid
598         Case Else
599             GoTo Invalid
600     End Select
601     Exit Function
602
603 Invalid:
604     BrowseForFolder = False
605 End Function
606
607 'reads the value for the registry key i_RegKey
608 'if the key cannot be found, the return value is ""
609 ' https://vba-corner.livejournal.com/3054.html
610 Function RegKeyRead(i_RegKey As String) As String
611     Dim myWS As Object
612
613     On Error Resume Next
614     'access Windows scripting
615     Set myWS = CreateObject("WScript.Shell")
616     'read key from registry
617     RegKeyRead = myWS.RegRead(i_RegKey)
618 End Function
```

3.1 Introduction

Personal reference for my own VIM environment setup and scripts.

Repository on GitHub: [vimfiles](#).

3.2 GVim RC

```
1 colorscheme darkblue
2 set guifont=Lucida_Console:h9:cANSI:qDRAFT
3 set hidden
4 set fileencodings=latin1,utf-8,ucs-bom
5 set encoding=latin1
6 set lines=40 columns=128
7 set number
8 set visualbell
9
10 command Edgvimrc e $HOME/Dropbox/_VIM/_gvimrc
11
12 source $HOME/Dropbox/_VIM/_gvim_simon
```

3.3 GVim Personal Setup

```
1 " vim: set syntax=vim:set encoding=latin1
2
3 source $HOME/SJH/scientia/vimfiles/_vim_common_simon
4
5 set listchars=eol:␣,tab:»>,trail:.,nbsp:.,extends:.,precedes:
```

(continues on next page)

(continued from previous page)

```
6 set list
7 set showbreak=|
```

4.1 Introduction

These pages are firstly “notes to self” on setting things up, and will reference private git repositories.

Perhaps, as this matures, I will clean it up for wider consumption.

4.2 MT4 Setup

- Install MT4 from the install file provided by the broker

Modify the destination directory with a (n) suffix to install multiple copies on the same machine

- Git clone the MT4 base repo into a new folder on the machine, then manually copy it across to overlay onto the MQL4 folder in Terminal datadirectory.

```
git clone https://github.com/simon-hoffe/mql5-mt4-base.git
git clone git@github.com:simon-hoffe/mql5-mt4-base.git
```

Then need to initialise the sub-modules and clone them as well.

```
git submodule update --init --recursive
```

- genindex

M

mql5, 16
mt4, 16
mt5, 16

O

outlook, 1

V

vba, 1
vim, 14