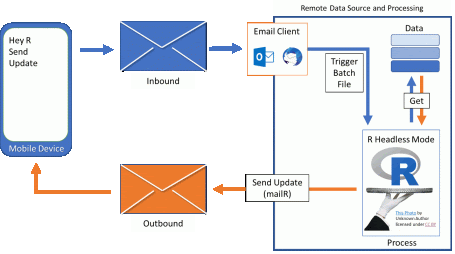
Say hello to the MobileTrigger package. With it, you can run R scripts, models, and reports from a mobile device using e-mail triggers. In its debut offering, version 0.0.31 of the package provides List and Run trigger functions. The list function allows you to see available scripts, models, and reports. Complimenting list, the run function allows you to launch and interact with R while you're traveling and cannot be at your desk to run R directly. In this post, we'll cover MobileTrigger installation and setup. Follow-up articles will detail how to set up scripts, reports, and models for mobile execution and interaction.

**The Concept**

Let’s, I showed you how a mobile device could be used to trigger an R script back at your home or office. In a nut shell, The process looked like this:



1. Write a **R script** on the remote machine that *performed an operation* and *e-mails the results back* to your mobile device using the mailR package.
2. Create a special batch file that calls “Headless R” and your target **R script**.
3. Set your desktop e-mail client to *launch* the special batch file when an e-mail having the subject “Hey R – Send Update” is received from your mobile device

Run an R script in Headless Mode

The Rscript we are going to run headless is our E-mail script described above. Here is the complete script if you haven't saved it already:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | # Parse the HTML  **require**(rvest) #package to parse HTML  **page** <- read\_html("https://apps.r-bar.net/PID1025/") #Get the Page  tbls <- html\_nodes(**page**, "table") #Get the Tables  Product\_Status <- html\_table(tbls[1])[[1]] #Get First Table    #Detemine how much product is shipping today as of right now  ToShip <- **grep**(pattern = "Shipping Yesterday|Shipping Today",  x = Product\_Status$`Product Metrics`)  Shipping\_Product <- **sum**(Product\_Status$Count[ToShip])    #Detemine how many minute before the shipping dead line  MinLeft <- **gsub**(pattern = '**\\**D',  replacement = "" ,  Product\_Status$`Product Metrics`)[[2]]    #The summary message to be e-mailed  email.msg <-  paste0(Shipping\_Product, " units of product shipping as of ", **date**(),  ". There is ", MinLeft,  " minutes remaining before today's shipping deadline.")    #Email Trigger when "Hey R - Send Update"  **library**(mailR)  sender <- "mydesktop@work.com" #The email address R will use to send  recipients <- **c**("myMobile@home.com") #your mobile email here  send.mail(from = sender,  to = recipients,  subject = "Requested Status Report",  **body** = email.msg, #your summary message  smtp = **list**(host.name = "your.smtp.server",#check with IT or ISP  port = 587, #port number (IT or ISP)  user.name = "your.username", #likely your email address  passwd = "your.password", #email password  ssl = TRUE #if using secure connection  ),  authenticate = TRUE,  send = TRUE) |

## **E-mailing the Data from R to our Mobile Device**

Awesome, we have the data in R and we generated the message we want to send. Now, we need to send it to our mobile device using R. To make this work, you will need to have the following information on hand:

* SMTP server for outgoing e-mail
* The SMTP server port
* Username
* Password

If you are not sure about these items, check with IT or your internet service provider (ISP). Using this information and R's mailR package, we can generate some code to send our message from the previous section to our mobile device via e-mail. Here is some example code.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #Email Trigger when "Hey R - Send Update"  **library**(mailR)  sender <- "mydesktop@work.com" #The email address R will use to send  recipients <- **c**("myMobile@home.com") #your mobile email here  send.mail(from = sender,  to = recipients,  subject = "Requested Status Report",  **body** = email.msg, #your summary message  smtp = **list**(host.name = "your.smtp.server",#check with IT or ISP  port = 587, #port number (IT or ISP)  user.name = "your.username", #likely your email address  passwd = "your.password", #email password  ssl = TRUE #if using secure connection  ),  authenticate = TRUE,  send = TRUE) |

## **Pre-Processing the Data for Our Mobile Report**

Great, we have a data source and we are pretending that it can't be directly accessed with our mobile device. But, we can process it with our desktops using R. So, suppose we are on the road, and we really need to know the following: **As of right now, how much product will be shipping today and how much time is there before the shipping deadline at the New York site?**

Using our desktops, we generate the following code which i) scrapes the data from the data source, ii) sums the appropriate data, and iii) makes a short text summary:

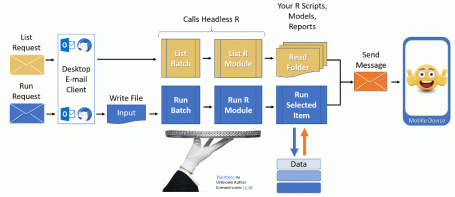
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | # Parse the HTML  **require**(rvest) #package to parse HTML  **page** <- read\_html("https://apps.r-bar.net/PID1025/") #Get the Page  tbls <- html\_nodes(**page**, "table") #Get the Tables  Product\_Status <- html\_table(tbls[1])[[1]] #Get First Table    #Detemine how much product is shipping today as of right now  ToShip <- **grep**(pattern = "Shipping Yesterday|Shipping Today",  x = Product\_Status$`Product Metrics`)  Shipping\_Product <- **sum**(Product\_Status$Count[ToShip])    #Detemine how many minute before the shipping dead line  MinLeft <- **gsub**(pattern = '**\\**D',  replacement = "" ,  Product\_Status$`Product Metrics`)[[2]]    #The summary message to be e-mailed  email.msg <-  paste0(Shipping\_Product, " units of product shipping as of ", **date**(),  ". There is ", MinLeft,  " minutes remaining before today's shipping deadline.") |

**The MobileTrigger Package**

Having the ability to launch a single R script remotely is nifty. But, most of us need to run many R scripts, models, and reports during the day.

hmmm… How to handle this?

To work more generally with R via e-mail, I started working on the MobileTrigger package which is now on CRAN. The package allows for basic two-way communication between your mobile device and R. As discussed above, communication relies strongly on e-mail client rules, batch files, and headless R. The graphic below summarizes the core workflow of the MobileTrigger package.



**Package Note:** At this time, the MobileTrigger package only works for Windows users who read their e-mail with either Outlook or ThunderBird. I'm confident a solution could be made for Mac and Linux environments. If you would like to contribute to the project.

Now that we have oriented ourselves, let's install and setup the MobileTrigger package.

**Install MobileTrigger**

Mobile triggers can be installed from CRAN or GitHub using the following code:

|  |  |
| --- | --- |
| 1  2 | install.package("MobileTrigger") # CRAN |

**Planning Your MobileTrigger Setup**

There is a lot of communication going on in the MobileTrigger package. Specifically communication needs to happen between the following three systems: your **Mobile**, **R**, and your **desktop e-mail client**. You can do this using **the same** e-mail address for all systems, **a different** e-mail address for each system, or **a hybrid** as detailed below:

**MobileTriggers Uses One E-mail Address**

|  |  |
| --- | --- |
| 1 | MobileEmail <- RsEmail <- MailClientEmail <- "Your.Email@somewhere.com" |

The benefit to using a single e-mail address is that you only need SMTP server settings for that one e-mail address.

**MobileTriggers Uses Two E-mail Address**

|  |  |
| --- | --- |
| 1  2 | MobileEmail <- "Email@MobileDevice.com"  RsEmail <- MailClientEmail <- "Your.Email@somewhere.com" |

For this you will need SMTP settings for both e-mail addresses.

**MobileTriggers Uses Three E-mail Address**

|  |  |
| --- | --- |
| 1  2  3 | MobileEmail <- "Email@MobileDevice.com"  RsEmail <- "For.R@R-email.com"  MailClientEmail <- "Your.Email@somewhere.com" |

For this you will need SMTP settings for both R and the Mobile e-mail addresses.

In the next sections that follow we are going to use the three e-mail address approach. If you understand how to work with three email addresses than choosing to setup the mobileTrigger package using the 1 or 2 e-mail address options is trivial.

**MobileTrigger File and Folder Setup**

To get us rolling with the MobileTrigger package, we are going to start with the SetupWindowsTrigger() function. The function creates most of the files you need to facilitate communication between R and your mobile device. To use SetupWindowsTrigger(), you need to know the e-mail sever details you want to use **for R to send messages** to your mobile device. Please make sure you have the following information on hand:

* SMTP server for outgoing e-mail
* The SMTP server port
* Username
* Password

Other pieces of information you will need for the SetupWindowsTrigger() function include the

* **“root trigger path”:** The root trigger path is where MobileTrigger will conduct its business. It will also be the home of the scripts, models, and reports you plan to use with MobileTrigger. I recommend you put it some place simple like “c:/trigger” if you can.

**Note:** At this time **spaces** in the trigger path is poorly supported. Recommend that you keep the trigger path simple (e.g., “c:/trigger”)

* **Mobile E-Mail Address:** The package uses this to send results to your mobile device. Specify this e-mail address using the function’s *Mail.To* argument.
* **R’s Sending E-mail Address:** R will need a way to communicate out using the mailR package. You may want to make an e-mail address like R@mydomain.com. Specify this e-mail address using the function’s *Mail.From* argument

Below is an example call to the SetupWindowsTrigger() function showing how all this information gets used.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | **require**(MobileTrigger)    ################################################  # Define path & E-mail Addresses #  ################################################  path <- "c:/trigger"  MobileEmail <- "Email@MobileDevice.com"  RsEmail <- "For.R@outlook.com"  MailClientEmail <- "Your.Email@somewhere.com"    ###################################################  # R's Email Outlook SMTP Example (uses tls= TRUE) #  ###################################################  SetupWindowsTrigger(  path = path,  Mail.To = MobileEmail,  Mail.From = RsEmail,  SMTP.Settings = **list**(host.name = "smtp.office365.com",  port = 587,  user.name = RsEmail,  passwd = 'password',  tls = TRUE)    ) |

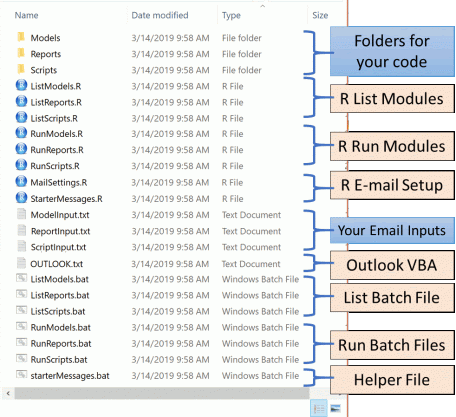
## **Generic mailR SMTP Settings (SSL and TLS)**

To send e-mails with the mailR package, you use the send.mail() command. Here is a generic example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | **require**(mailR)  send.mail(  from="[from email address]", to = "[to email address] ",  subject = "The Subject line",  **body** = "The message you want to send",  smtp = **list**(host.name = "some.smtp.server.com",  port = [PortNumber],  user.name = "[from email address]",  passwd = "[YourPassword]",  #ssl = T  #tls = T),  authenticate = **T**, send = **T**  ) |

**After Running SetupWindowsTrigger**

After running the SetupWindowsTrigger function, take a moment to examine what's in the root trigger path. Below is a screenshot of what mine looks like with some descriptions.



As you can see from the screenshot above, the folder contains e-mail input files, a series of batch and R files for headless R operations, and some folders to put your codes depending on what you're trying to accomplish. All these files work together to facilitate communications between R and your mobile device.

If your trigger folder is written and you have a basic understanding of the files and folders, let's start setting up your e-mail client to work with the MobileTrigger framework.

**Setting up the e-mail client**

This section will help you get your desktop e-mail client (Outlook or ThunderBird) ready to work with the MobileTrigger package. This is the most involved part of the setup, but once you're done it should be trouble free.

**ThunderBird**

There are two ways to setup ThunderBird with the MobileTrigger framework, the fast way and the slow way. In general the process goes like this:

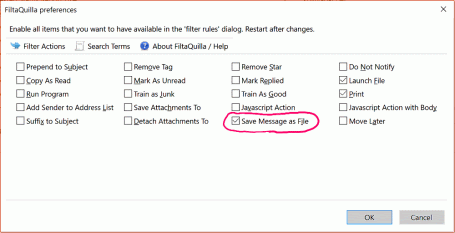
1. Install FiltaQuilla Plugin
2. Adjust FiltaQuilla options so the “Save Message as File” option to ON
3. Setup ThunderBird Rules
4. Test Communication

In the next few sections, we'll discuss the ThunderBird setup in detail. We'll start with the fast way; then I'll show an example of the slow way. Let's get started by installing the FiltaQuilla add-on.

**Install FiltaQuilla add-on**

We need to install a plugin called FiltaQuilla, so we can make e-mail filters in ThunderBird. To install the plugin:

1. Open ThunderBird and let everything sync-up
2. Press (ALT + t) on your keyboard. At the top of the ThunderBird window, the tools menu should appear
3. Click the “Add-ons” menu item. This will open a new tab within ThunderBird called the “Add-ons Manager”.
4. Click “Extensions” in the left-hand panel to open the extensions screen. Use the search box in the top-right corner to search for FiltaQuilla
5. Install it
6. Restart ThunderBird when you’re done.
7. After restarting, press (ALT + t) again. This time select “Add-on Options” → “FiltaQuilla”. In the dialogue window, make sure the “Save Message as File” option is checked. Press OK. (see image below)



FiltaQuilla is now installed and ready. Close ThunderBird

**Setup ThunderBird Rules: The Fast Way**

This process takes me about 2 minutes. That being said, I've had a lot of developer practice. Anyway, the method we are about to do is much faster than the “slow” ThunderBird method or the Outlook method. Why? Because Filtaquilla's e-mail rules are in a nice editable text file called:

msgFilterRules.dat

1. Make sure ThunderBird is closed
2. Find the file **msgFilterRules.dat** buried deep in your windows home directory. For me, the path looked a little something like this:

C:/Users/[username]/AppData/Roaming/ThunderBird/Profiles/wrvfw5me.default/ImapMail/imap.ionos.com/msgFilterRules.dat

The highlighted portion of the path above is likely to be a little different for your local system, but the above should give you an indication of what the path looks like.

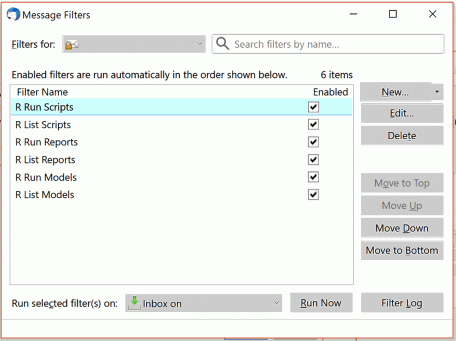
1. Open the **msgFilterRules.dat** file in your favorite text editor. Initially, I had these two lines of text:

|  |  |
| --- | --- |
| 1  2 | **version**="9"  logging="no" |

1. If you already have filters working in your environment, backup the file before experimenting with it. Mistakes happen!
2. In R, run the WriteThunderBirdFilters() function from the MobileTrigger package:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | **require**(MobileTrigger)    ################################################  # Define path & E-mail Addresses #  ################################################  path <- "c:/trigger" #your root trigger path  MobileEmail <- "Email@MobileDevice.com"  RsEmail <- "For.R@outlook.com"  MailClientEmail <- "Your.Email@somewhere.com"    ################################################  # Write the ThunderBird Filters #  ################################################    WriteThunderBirdFilters(  path = path,  sent.from = MobileEmail  ) |

1. After running this file, a new **msgFilterRules.dat** will be written to your root trigger path (in our examples the root trigger path has been c:/trigger).
2. If you just installed FiltaQuilla for the first time, you can overwrite the existing **msgFilterRules.dat** file with the new one generated by R. If you have existing rules, copy the text from the R generated file from line 3 onward. Paste-append that text to the end of your existing **msgFilterRules.dat** using a text editor. Save and close.
3. Open ThunderBird
4. Press (ALT + t) to open the program tools menu and select the “Message Filters” menu item. The dialogue box that appears should show 6 rules as illustrated below:



1. To check your setup, skip down to the **Communication Testing** section.

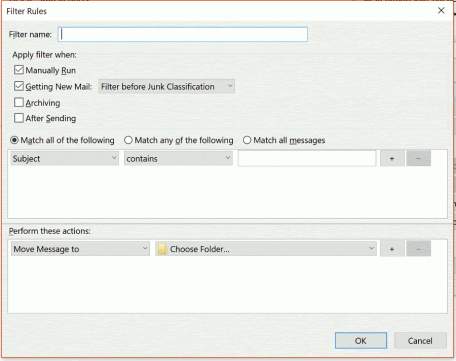
**Setup ThunderBird Rules: The Slow Way**

If you're in this section, you have FiltaQuilla installed, ThunderBird is presently closed, and you have elected to setup the rules using the slow way. This will take about 15-20 minutes. If you don't want to set up the rules this way, go to the [Fast Way](https://r-bar.net/mobiletrigger-setup-run-r-scripts-mobile-device/#TH_Fast_Way) section.

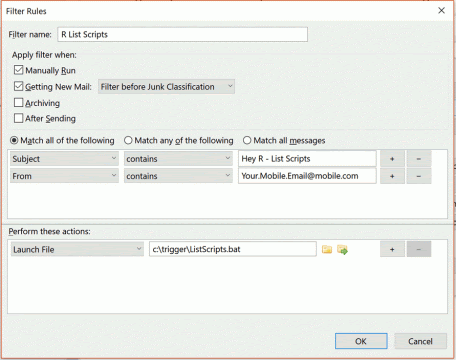
Alright, the slow way.

**Example List Script Rule**

1. Open ThunderBird
2. Go to your root trigger path. In our running example, this would be c:/trigger.
3. Run the “starterMessage.bat” file. This will launch headless R, and use the e-mail settings you’ve provided when you built the folder to send 6 messages with different subjects. Specifically, “Hey R – List Scripts”, “Hey R – Run Scripts”, “Hey R – List Models”, “Hey R – Run Models”, “Hey R – List Reports”, “Hey R – Run Reports”.
4. When you receive the messages, copy the subject line from the “Hey R – List Scripts” email.
5. Press (Alt + t), to open the ThunderBird Tool Menu, then click “Message Filters” and Press “New”. The following screen will appear.



1. Name the rule “R List Scripts”, and specify a rule that responds to two conditions
   1. subject line is “Hey R – List Scripts”
   2. message is from your mobile e-mail address
2. Next, set the rule action to launch the ListScripts.bat file in your trigger root folder. In our examples, we’ve been using c:\trigger. The resulting e-mail rule should look like the screenshot below. Other list types will look similar:



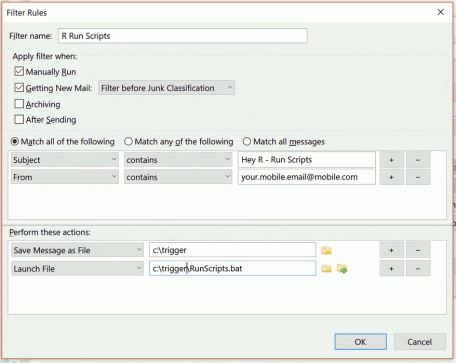
1. After pressing OK, restart ThunderBird
2. Test: Send an e-mail to your desktop with the subject line “Hey R – List Scripts” from your mobile device to make sure the batch file gets launched. You should see a command prompt open briefly with some R output. The message that returns to your mobile, will indicating “No Scripts in Path”. This is expected

We just setup the “List Script” trigger. Now, we need to setup the “Run Script” e-mail trigger.

**Example Run Script Rule**

1. Copy the subject line from the “Hey R – Run Scripts” message.
2. Open the FiltaQuilla Rule Manager using (Alt + t) → “Message Filters”
3. Setup the run script rule as shown in the image below. Notice the filter criteria is the same as before, but the action has two operations:
   1. write message to file in the root trigger path (C:/trigger)
   2. Launch RunScripts.bat in the root trigger path

Completed rule should look like this:



1. Press OK
2. Restart ThunderBird
3. **Test:** Send an e-mail to your desktop with the subject line “Hey R – Run Scripts” from your mobile device. Make sure the batch file gets launched. During the process, you should see a command prompt open briefly with some R output. Shortly after, you should get an e-mail message indicating “No Scripts in Path” sent back to your mobile. This is expected

If the first two triggers appear to be working as expected, congratulations! You made it through a very long process. To finish the process, grab a coffee, put on some upbeat music, and repeat the above procedure on the last 4 triggers listed in the table below as appropriate.

| **Rule Name** | **Filter Action** | **Filter Criteria** |
| --- | --- | --- |
| List Scripts | Launch: [Trigger Path]/ListScripts.bat | Subject == ‘Hey R – List Scripts’ From == ‘your.mobile.email@mobile.com’ |
| Run Scripts | 1. Save Message as File to [Trigger Path]/ 2. Launch: [Trigger Path]/RunScripts.bat | Subject == ‘Hey R – Run Scripts’ From == ‘your.mobile.email@mobile.com’ |
| List Models | Launch: [Trigger Path]/ListModels.bat | Subject == ‘Hey R – List Models’ From == ‘your.mobile.email@mobile.com’ |
| Run Models | 1. Save Message as File to [Trigger Path]/ 2. Launch: [Trigger Path]/RunModels.bat | Subject == ‘Hey R – Run Models’ From == ‘your.mobile.email@mobile.com’ |
| List Reports | Launch: [Trigger Path]/ListReports.bat | Subject == ‘Hey R – List Reports’ From == ‘your.mobile.email@mobile.com’ |
| Run Reports | 1. Save Message as File to [Trigger Path]/ 2. Launch: [Trigger Path]/RunReports.bat | Subject == ‘Hey R – Run Reports’ From == ‘your.mobile.email@mobile.com’ |

After you get all the email rules in, you'll want to test to make sure everything triggers as planned. You may have been doing this as you setup each trigger. If so, move on to the **next steps** section, otherwise skip down to the **Communication Testing** section.

## **Communication Testing**

Congratulations, if you've made it to this section I assume you have all the rules setup in either Outlook or ThunderBird. Perhaps, you've tested your communications as you set up each rule. If you know the rules are working, skip this section and move on to this post's **Summary**. If you still need to test your setup, here is an R script to help you through the process. The script is organized as follows.

1. Load the Required Libraries
2. Run the trigger tests

The testTrigger() function in the script below will make requests to List and Run scripts, models, and reports that don't exist currently in your setup. This is fine. We are checking to make sure communication is working. In follow-up posts, you'll get to see these modules in action.

To test your triggers and communications, start by running the “list” triggers line [9]. There is a 15 second delay between each message sent. As the requests roll in, your triggers should activate, and you should start seeing e-mail from R indicating “no” model, script, or report in path. If these are working, move on to test each set of Run Triggers (Model, Script, and Report). Remember to update the Mail.To and Mail.From for your situation.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | # Load Libraries ----------------------------------------------------------  **require**(mailR)  **require**(MobileTrigger)    ################################################  # Define path & E-mail Addresses #  ################################################  path <- "c:/trigger" #your root trigger path  MobileEmail <- "Email@MobileDevice.com"  RsEmail <- "For.R@outlook.com"  MailClientEmail <- "Your.Email@somewhere.com"    ################################################  # Test Communication #  # - Test Each of the TestWhat Conditions #  ################################################    testTriggers(  TestWhat = "Lists",  #TestWhat = "RunModels",  #TestWhat = "RunScripts",  #TestWhat = "RunReports",  path = path,  Mail.To = MailClientEmail, # Client Email Address  Mail.From = MobileEmail, # Simulate Mobile  Mail.From.SMTP.Settings =  **list**(host.name = "Mobile SMTP Server Settings",  port = [mobile SMTP port number],  user.name = MobileEmail,  passwd = 'Mobile Password',  tls = TRUE)  ) |

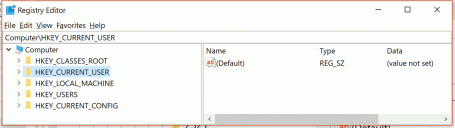
**Outlook**

Setting MobileTrigger up with Outlook is going to take about 20 minutes. The process goes roughly like this:

1. Tweek the Windows Registry
2. Open Outlook
3. Setup VBA Scripts & Enable Macros
4. Restart Outlook
5. Send Starter Messages
6. Make the Rules
7. Address Outlook Bug
8. Test Communication

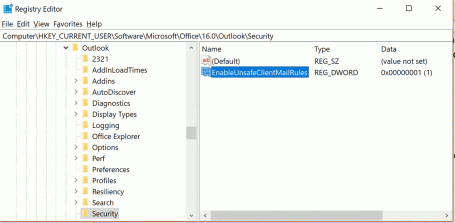
**Registry Tweak**

1. Press the (Windows Key + R): This will open the run dialogue box
2. In the “open” text box type: regedit
3. You’ll get an “are you sure you want to use this application” warning from windows. Press OK…
4. The registry editor should look like this:



Cool we got the registry editor open. Here are the changes we need to make

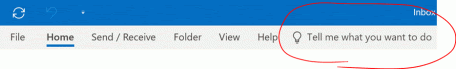
1. In the left-hand panel drill down to:  
   **Computer\HKEY\_CURRENT\_USER\Software\Microsoft\Office\16.0\Outlook**
2. Make a new folder (key) called “Security” by right clicking on the Outlook folder and selecting “New”→“Key”
3. Enter the new “Security” folder. Full path should now be:  
   **Computer\HKEY\_CURRENT\_USER\Software\Microsoft\Office\16.0\Outlook\Security**
4. Create a new DWORD(32bit) value in the right panel by right clicking, call it “EnableUnsafeClientMailRules”
5. Double click on EnableUnsafeClientMailRules and set the data value to 1.
6. Should look like this when you are done:



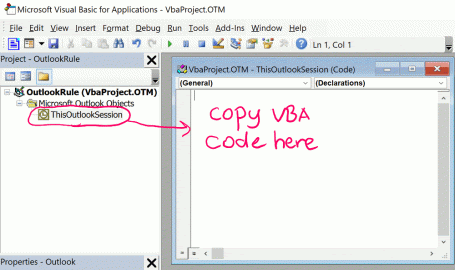
All done.

**Setup the Visual Basic Macros**

1. In the root path for MobileTrigger, open the file OUTLOOK.txt in a text editor and copy the text. The contents of this file is visual basic for applications (VBA) code. We need this to setup our email rules.
2. Open Outlook
3. On the program menu find the “Tell me what do search box”



1. Search for “Visual Basic Editor” and open it. A new window should popup
2. Find the “thisOutlookSession” module circled in the screenshot below and double click. Likely an empty window will appear. See example image below

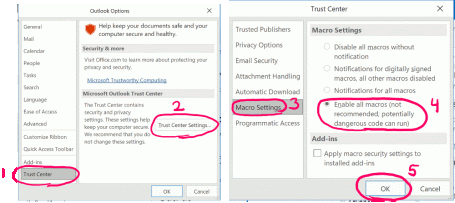


1. Paste in the VBA code you copied form the OUTLOOK.txt file (see step 1).
2. Save and close the Visual Basic Editor

**Enable Macros**

If you don't enable macros, the MobileTrigger package won't work with Outlook. If the security risk is too great for you, consult your IT department or perhaps consider using ThunderBird instead.

1. Go to “File” → “Options”
2. Select the Trust Center
3. Click the Trust Center Settings (a new window will appear)
4. Select the Macro Settings Option
5. Adjust the macro settings to “Enable All Macros”. **Note** Notification of all macros will work but only within a given Outlook session.



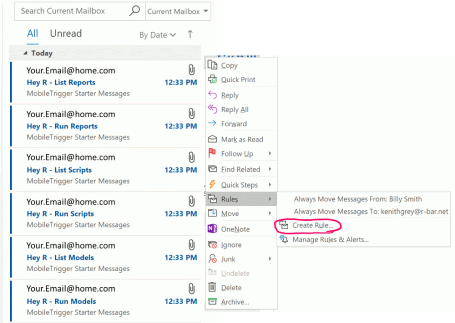
1. Press OK.
2. Restart Outlook.

**Starter Messages**

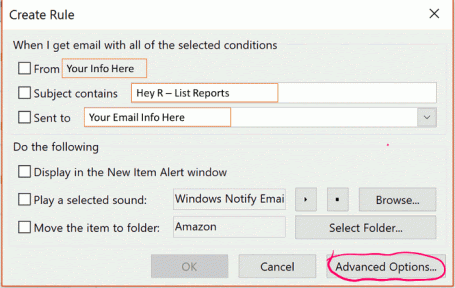
1. Run the starterMessages.bat file in your root trigger path
2. 6 messages should appear in your Outlook e-mail box.

**Setup the Rules**

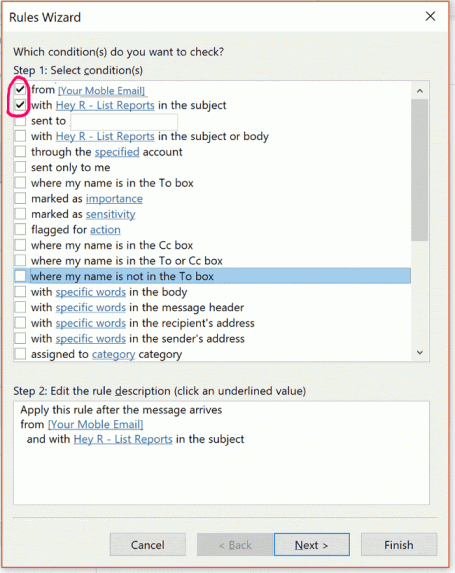
1. Right Click on the e-mail with subject “Hey R – List Reports”
2. Go to Rules → “Create Rule…”



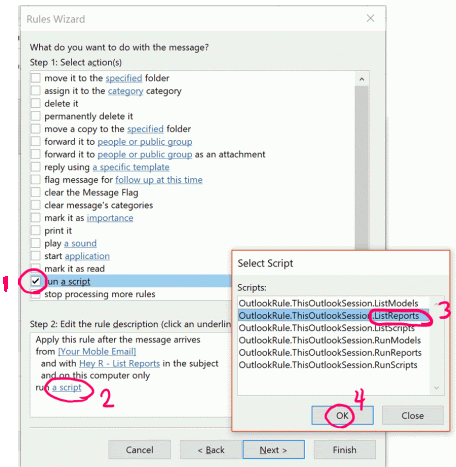
1. Select advanced from the Create Rule Dialogue Box



1. **First screen** – Set rule conditions
   1. Select the “From [Your Mobile e-mail]” and
   2. With “Hey R – List Reports” in the subject
   3. Press Next



1. **Second screen** – set the action to do if the previous conditions were met
   1. In the Step 1: Select action box, select “run a script”. This option won’t be available if you haven’t made the windows registry change discussed above.
   2. In the Step 2: Edit the Rule Description… Click the words “a script”
   3. A “Select Script” dialog box will appear. Select the VBA macro for listing reports, and press “OK”



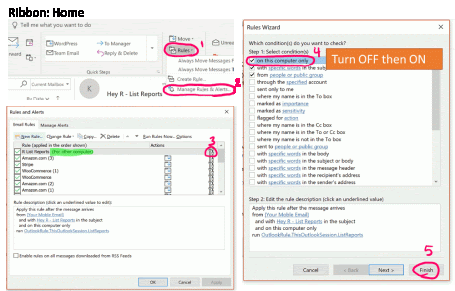
* 1. Press “Next” to move to the third screen

1. **Third screen** – set exceptions to the rules. We aren't going to worry about this right now.
   1. Press Next
2. **Last Screen** – Finish Rule
   1. Step 1: Give your Rule a name
   2. Step 2: Make sure the rule is turned on
   3. Press Finish

**Address the Outlook Bug**

If you try to test the rule at this point, it may not work because of an Outlook bug. This bug tripped me up for several hours. So hopefully this saves you some time.

* 1. On the Home Ribbon, click “Rules” → “Manage Rules & Alerts”
  2. A Dialogue box will open, and show your new rule for R to list reports. To the right of the rule name you will see “(For Other Computer)”. For whatever reason this causes problems.
  3. To get rid of it, click the wrench and screw driver at the far right (see picture below).
  4. The Rule Wizard will open, Toggle OFF then ON “on this computer”



* 1. Press Finish
  2. You should no longer see the “(For Other Computer)” in your rule list.
  3. Press OK
  4. Check that the List Report Rule is working by sending a test e-mail to Outlook with the header “Hey R – List Reports”. Make sure its from the right e-mail address or the rule won’t run.

If the test works, repeat the above steps on the other 5 triggers. Here is a complete list.

| **Rule Name** | **Use Script** | **Filter Criteria** |
| --- | --- | --- |
| List Scripts | OutlookRule.ThisOutlookSession.ListReports | Subject == ‘Hey R – List Reports’ From == ‘your.mobile.email@mobile.com’ |
| Run Scripts | OutlookRule.ThisOutlookSession.RunReports | Subject == ‘Hey R – Run Reports’ From == ‘your.mobile.email@mobile.com’ |
| List Models | OutlookRule.ThisOutlookSession.ListScripts | Subject == ‘Hey R – List Scripts’ From == ‘your.mobile.email@mobile.com’ |
| Run Models | OutlookRule.ThisOutlookSession.RunScripts | Subject == ‘Hey R – Run Scripts’ From == ‘your.mobile.email@mobile.com’ |
| List Reports | OutlookRule.ThisOutlookSession.ListModels | Subject == ‘Hey R – List Models’ From == ‘your.mobile.email@mobile.com’ |
| Run Reports | OutlookRule.ThisOutlookSession.RunModels | Subject == ‘Hey R – Run Models’ From == ‘your.mobile.email@mobile.com’ |

Once you have everything setup, move on to the **Communication Testing** section

**Communication Testing**

Congratulations, if you've made it to this section I assume you have all the rules setup in either Outlook or ThunderBird. Perhaps, you've tested your communications as you set up each rule. If you know the rules are working, skip this section and move on to this post's **Summary**. If you still need to test your setup, here is an R script to help you through the process. The script is organized as follows.

* 1. Load the Required Libraries
  2. Run the trigger tests

The testTrigger() function in the script below will make requests to List and Run scripts, models, and reports that don't exist currently in your setup. This is fine. We are checking to make sure communication is working. In follow-up posts, you'll get to see these modules in action.

To test your triggers and communications, start by running the “list” triggers line [9]. There is a 15 second delay between each message sent. As the requests roll in, your triggers should activate, and you should start seeing e-mail from R indicating “no” model, script, or report in path. If these are working, move on to test each set of Run Triggers (Model, Script, and Report). Remember to update the Mail.To and Mail.From for your situation.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | # Load Libraries ----------------------------------------------------------  **require**(mailR)  **require**(MobileTrigger)    ################################################  # Define path & E-mail Addresses #  ################################################  path <- "c:/trigger" #your root trigger path  MobileEmail <- "Email@MobileDevice.com"  RsEmail <- "For.R@outlook.com"  MailClientEmail <- "Your.Email@somewhere.com"    ################################################  # Test Communication #  # - Test Each of the TestWhat Conditions #  ################################################    testTriggers(  TestWhat = "Lists",  #TestWhat = "RunModels",  #TestWhat = "RunScripts",  #TestWhat = "RunReports",  path = path,  Mail.To = MailClientEmail, # Client Email Address  Mail.From = MobileEmail, # Simulate Mobile  Mail.From.SMTP.Settings =  **list**(host.name = "Mobile SMTP Server Settings",  port = [mobile SMTP port number],  user.name = MobileEmail,  passwd = 'Mobile Password',  tls = TRUE)  ) |

If all “List” and “Run” modules are communicating, MobileTrigger is setup and ready for use.

**Next Steps**

If MobileTrigger is working with your e-mail client, the next step is to bring in your models, scripts, and reports. We will cover each case in a series of brief follow-up posts. So, stay tuned!

**Summary**

You made it to the end. Yes, I know setting up the e-mail client is a pain. However, if you've made it this far you've got the basic idea of how the MobileTrigger package works. As well, you should have working e-mail communications between your mobile device and R on a remote desktop at your home or office. In a series of **brief** upcoming posts, I will show you how to integrate your reports, models, and scripts.