Integrating Email and Text Messaging Status Updates with Global Meteor Network's Raspberry Pi Meteor Station

Version 0.6.1

26 April 2022

While the operation of the Global Meteor Network's (GMN), Raspberry Pi Meteor Station (RMS) software is fully automated and mostly autonomous, it is desirable to be able to monitor the health of the system and alert the camera owner and / or network administrator of issues affecting operations.

This document describes one such implementation which allows for sending status reports via email and / or text messages. It involves the following steps:

1. Establish dedicated email address for use in sending messages via smtp.
2. Install smtp local server on the RPi associated with the camera using the PostFix software package.
3. Implement Python script to create and send email / text message containing desired GMN camera health status.
4. Automate the sending of above message.

```
****************************************************************************
*  NOTE: This implementation requires RPi4 – Buster OS. A RPI3 – Jessie version is currently   *
*  being tested. If successful this document will be updated to support RPI3.                  *
*                                                                                              *
****************************************************************************
```

Email account – Almost any email service may be used for the purpose of sending GMN camera status reports. While the process described herein is relatively secure it is probably best not to expose a personal or business account. The recommendation is to establish a dedicated email account / address.

To establish a new gmail GMN camera email address go to www.gmail.com and create a new email account. Recommend something similar to XX000Xgmn@gmail.com where XX000X represents your camera's assigned GMN id. **PLEASE NOTE:**

> Effective May 2022 Google is requiring two factor authentication (TFA) for any app using a Gmail account. The 'Less secure' option will no longer be available. In order to comply and use your new Gmail account with RMS you will need to create a Gmail Application Password. The benefit to this is that the email account password will no longer be sent over the internet or stored in the Python script. See the link below on installing PostFix. It describes setting up a Gmail Application Password.

Installing PostFix – Postfix is a Unix based mail server program that is compatible with the Raspberry Pi.

To make your Gmail account 2FA compliant and install the PostFix software on your Rpi please follow the excellent tutorial by Stacy Powell at Setting Up Gmail (and Other Email) on a Raspberry Pi | by Stacy Prowell | The Startup | Medium. NOTE: when prompted for 'System mail name' simply enter your station id (XX000X) or whatever your RPi HOSTName is. This is not critical.

Python script – The appendix contains the Python script listing. This script will establish the smtp client, retrieve the last row of the fits_count file, check the file_to_upload file for any files not uploaded and send this information via email and / or text message to the addresses of your choice. Copy / paste the code in the appendix into a plain text file using something like NotePad (Windows) or Nano (RPi / Linux) and save it as daily_status_report.py.

If you are not familiar with Python that is not a problem. All you need to do is set the parameters at the beginning of the script for your particular configuration:

```
""" Initialize parameters \

    Carrier sms gateways:\

      TMobile - @tmomail.net \

      Verizon - @vtext.com \

      ATT - @txt.att.net    """


send_email = True

send_text = False

from_addr = '??000?gmn@gmail.com'

to_addr = 'myaddress@gmail.com'

sms_gateway = '5551234567890@txt.att.net'

path_to_fits_counts_file = "/home/pi/RMS_data/csv/??000?_fits_counts.txt"

path_to_files_to_upload_file = "/home/pi/RMS_data/FILES_TO_UPLOAD.inf"
```

Place the daily_status_report.py script in the folder of your choice, typically

/home/pi/source/RMS/RMS

To test the script and your settings open a terminal window and type in the following command:

Python3 “/home/pi/source/RMS/RMS/daily_status_report.py”

If successful you should be immediately returned to the command prompt. Check your email and / or SMS messaging to see that the message is received.

To automate the running of the script you can either add it to any existing scripts you currently are running as part of the call from RMS to external script or you can create a cron job to run at a specific time of day, typically early morning after RMS completes its upload and reboot.

If you prefer you can wrap the Python script in a BASH script in a manner similar to the following:

File name = DailyStatusReport.sh

File content = python3 <path to python script location> daily_status_report.py

To call script: ./DailyStatusReport.sh

Once implemented this automated daily email / text message will both confirm your RPi RMS software is running and also give you some insight into how well it did the previous night. Additional information including attachments are certainly possible as well.

APPENDIX

Daily_status_update.py

Version 0.6.1

```
#daily_status_report
""" Retrieves: \
      1. last row of FITS Count file
      2. List of any files not yet uploaded to GMN Database \
   as email and / or text message.\
   Run as either cron job or call from RMS external script.\
   Expects smtp server installed locally.\
   version 0.6.1 - added check of FILES_TO_UPLOAD.inf.\
   version 0.6 - convert to using tail command"""

import smtplib
import os
import subprocess     # requires python 3.5 or higher

""" Initialize parameters \
      Carrier sms gateways:\
         TMobile - @tmomail.net \
         Verizon - @vtext.com \
         ATT - @txt.att.net    """

send_email = True
send_text = False
from_addr = '??000?gmn@gmail.com'
to_addr = 'someone@me.com', 'myaddress@gmail.com'
sms_gateway = '5551234567@txt.att.net'
path_to_fits_counts_file = "/home/pi/RMS_data/csv/??000?_fits_counts.txt"
path_to_files_to_upload_file = "/home/pi/RMS_data/FILES_TO_UPLOAD.inf"


""" Open the fits_count file and get last row"""
last_line = subprocess.run(['tail','-1', path_to_fits_counts_file], capture_output=True,
text=True)

""" Check whether any files remain to be uploaded"""
```

```
filesize = os.path.getsize(path_to_files_to_upload_file)
if (filesize == 0):
   upload_status = "    All files uploaded"
else:
   with open(path_to_files_to_upload_file, 'r') as upload_file:
      upload_status = "Files not uploaded:\r\n" + upload_file.read()


""" Open an instance of the smtp client"""

client = smtplib.SMTP('localhost')

""" Compose and send email and / or sms message """

status_msg = ("""\
   Subject: US000U Daily Status Report

   """ + str(last_line.stdout) + upload_status)

if send_email:
   client.sendmail(from_addr,to_addr,status_msg)

if send_text:
   client.sendmail(from_addr,sms_gateway,status_msg)
client.quit()
```