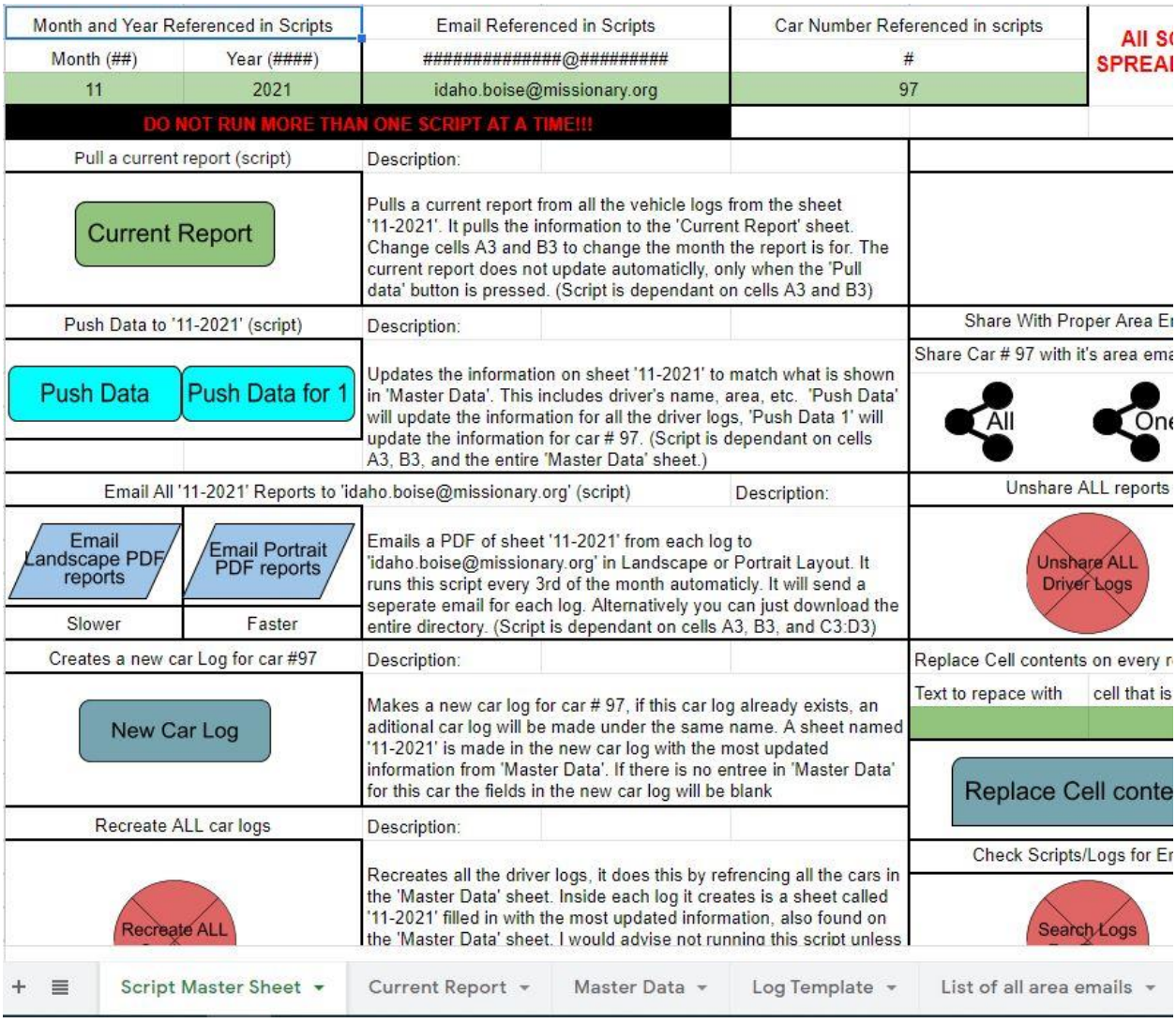
Virtual Mileage Logs Controller (VMLC) Manual



Designed, coded, and documented by Elder Colby James Ransom

Introduction

What is this Manual For?

This manual is to instruct you on how to use the Virtual Mileage Logs Controller or VMLC for short. This guide will help you understand all aspects of the VMLC and what they are for. This guide should explain to you what each thing is, why it is important, and how to use it along with other instructions.

Table of Contents

Chapter 1 ---------- General Guidelines – Page 5

- Chapter 1 – Part 1 – Running scripts
- Chapter 1 – Part 2 – What to do with errors
- Chapter 1 – Part 3 – Changing sheet names

Chapter 2 ---------- How to Get to VMLC – Page 7

- Chapter 2 – Part 1 – Signing into idaho.boise@missionary.org

Chapter 3 ---------- How to Maintain VMLC – Page 9

- Chapter 3 – Part 1 – Updating "Master Data"
- Chapter 3 – Part 2 – Updating "List of all area emails"
- Chapter 3 – Part 3 – Updating Cars mid-Month and mid-Transfer
- Chapter 3 – Part 4 – How to import Visual Basic macros into Excel

Chapter 4 ---------- Automated Actions Done for You – Page 19

- Chapter 4 – Part 1 – Creating Next Month's sheet
- Chapter 4 – Part 2 – Email reports to you on the 3rd of the month
- Chapter 4 – Part 3 – Missionary Email reminders on 1st and 3rd of the month
- Chapter 4 – Part 4 – Completed sheets are protected from editing
- Chapter 4 – Part 5 – How to disable/delete a trigger

Chapter 5 ---------- User Available Scripts - Page 25
Chapter 5 – Part 1 – Pull Data
Chapter 5 – Part 2 – Push Data/Push Data 1
Chapter 5 – Part 3 – Email PDF Report Landscape/Portrait
Chapter 5 – Part 4 – New Car Log
Chapter 5 – Part 5 – Recreate ALL Car Logs
Chapter 5 – Part 6 – Share with Proper Area Emails ALL/One
Chapter 5 – Part 7 – Unshare ALL Reports
Chapter 5 – Part 8 – Replace Cell Contents on Every Report
Chapter 5 – Part 9 – Check Scripts/Logs for Errors
Chapter 5 – Part 10 – Switch Cars

Chapter 6 ---------- Dealing with deprecated or obsolete code Page 34
Chapter 6 – Part 1 – Deciding what to do
Chapter 6 – Part 2 – Manually maintaining Driver Logs
Chapter 6 – Part 3 – Transitioning to Physical Logs or other.

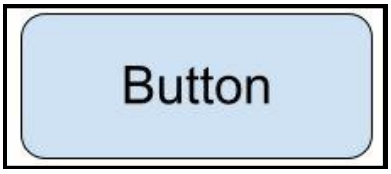
Chapter 7 ---------- Source Code For this Project Page 39
Chapter 7 – Part 1 – VMLC Source Google apps script
Chapter 7 – Part 2 – Excel Macro for converting Vehicles Assignment report
Chapter 7 – Part 3 – Excel Macro for getting the list of area emails

Chapter 8 ---------- Setup this Project for your Missions use Page 80
Chapter 8 – Part 1 – Guide for setup

Chapter 1 – General Guidelines

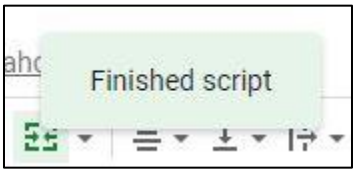
Part 1 – Running scripts

Running a script or function is accomplished by clicking on its associated button. The button starts the script.



After pushing a button you will see the "Running script" popup. This means the specified script is running and doing the work it was designed to do.

DO NOT RUN MORE THAN ONE SCRIPT AT A TIME! If you do there may be undesirable results, so just wait for the "Finished Script" popup before running other scripts, if any.

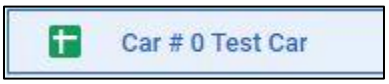


Part 2 – What to do with errors

While using this system, you may at times run into errors when running scripts. Errors appear like the "Running script" and "Finished Script" popups. The only difference is that instead of green, they show up in red.

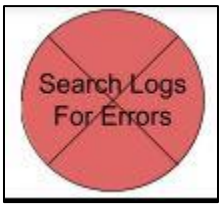
But what do I do about them? Depending on the error there may be several things that you will have to do to fix the problem. Generally, it's due to errors in the "Master Data" sheet such as "N/A" in the area emails column. That is usually fixed by updating the "List of all area emails" sheet. It's also good to check that the driver log you are working

with actually exists in the “Mileage Reports” folder and has a data entre in the “Master Data” sheet.



0	Test	Car

If everything so far looks good, the error may have been a temporary error, which sometimes happens. If you continue to get the error run the “Check Scripts/Logs for Errors” function.



It will search for more complicated problems. If it finds a problem, it will tell you how to fix it via a popup.

Part 3 – Changing sheet names

All the scripts are hard coded to reference specific sheets inside the Virtual Mileage Logs Controller (VMLC) by name. So, changing the names of these sheets will render almost all the scripts useless. The names of the sheets should be as follows (the order doesn’t matter) “Script Master Sheet”, “Current Report”, “Master Data”, “Log Template”, and “List of all area emails”. DO NOT CHANGE THESE NAMES.

Chapter 2 - How to Get to VMLC

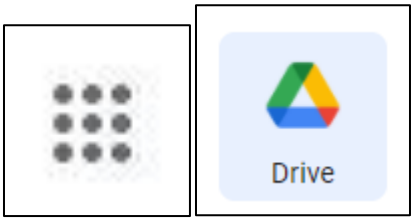
Part 1 – Signing into "idaho.boise@missionary.org"

(or your Office Google account)

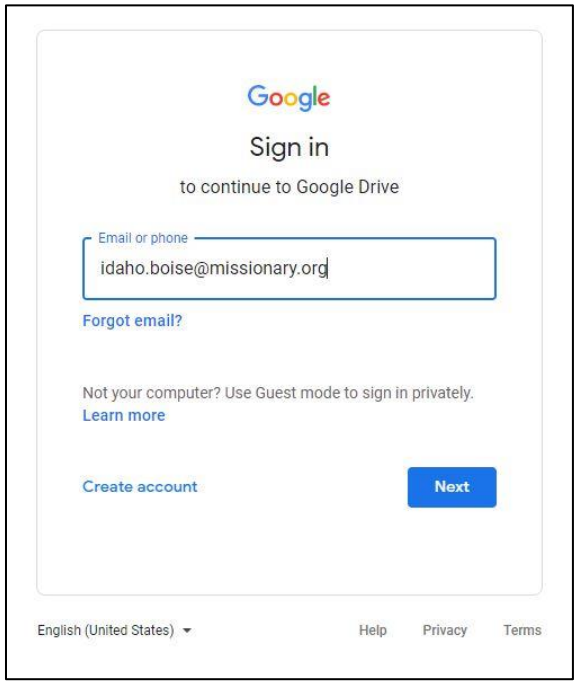
To access the "Virtual Mileage Logs Controller (VMLC)" sheet you will need to be signed into the "idaho.boise@missionary.org" account.

- Open Google Chrome if it is not already open.

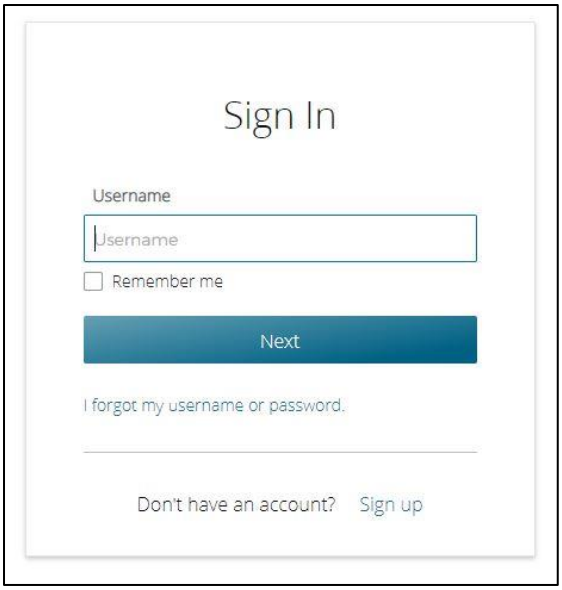
- Click on the nine dots and select google drive.



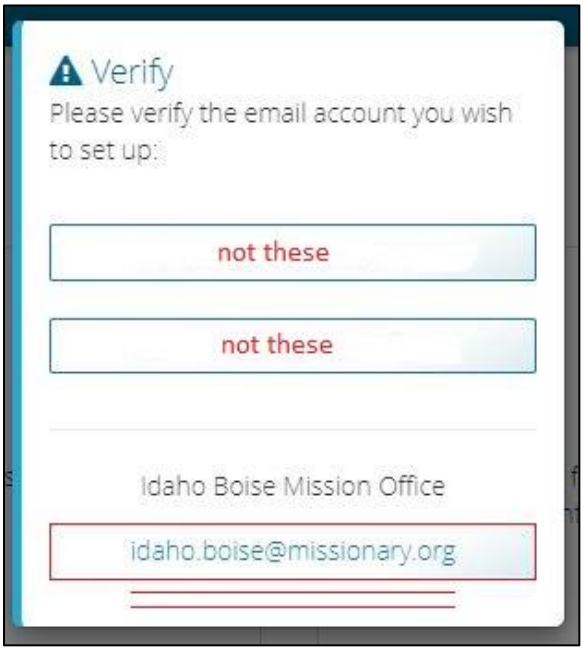
- You will then need to sign into the "idaho.boise@missionary.org" account. It will look something like this. If it does not look like this, add a google account, then navigate to google drive.



- It will then redirect you to the church login page.



- Once you log in to your church account, it will ask you what email you would like to sign into. Click the "idaho.boise@missionary.org" account.



- Once signed in, navigate to the inside of the "Mileage Reports" folder. (The mileage logs sent out to the missionaries are inside the "Mileage Logs Original" folder.) Note: There should not be any other folders present in this folder!

- Then open the "Virtual Mileage Logs Controller (VMLC)" spreadsheet.

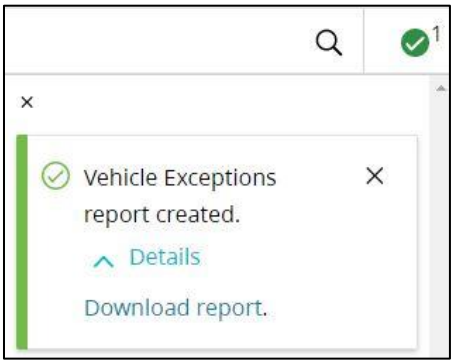
Chapter 3 - How to Maintain VMLC

Part 1 – Updating "Master Data"

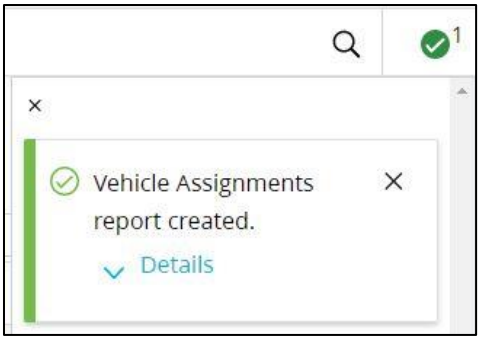
To keep "Virtual Mileage Logs Controller (VMLC)" up to date and working properly you will need to constantly update information in the "Master Data" sheet. This sheet contains all the information needed to keep the virtual mileage logs up to date. Its format is a variation of the "Vehicles Assignment" report pulled from IMOS. The steps to update the entire "Master Data" are as follows:

1. Log on to IMOS/the Missionary Portal
2. Navigate on the left-hand side panel and look for "Reports and Surveys"

3. Click on "Mission Office Reports"
4. Download the "Vehicle Exceptions" report.



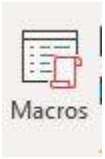
5. Check the "Vehicle Exceptions" report for any "Designated Driver Errors" and "Cars Not Assigned to an Area". Any cars that are not assigned to an area will not show up in the next report we pull from IMOS, so be sure all the cars are assigned to an area. Otherwise, those cars will have to be manually added to "Master Data" later.
6. Next, we need to pull the "Vehicle Assignments" Report. Refollow steps 1-3 then continue to the next step.
7. Download the "Vehicle Assignments" report in the Excel format.



8. Open the “Vehicle Assignments” report in Excel.
9. Turn off “Protected View”

10. Switch to the Developer Tab. If you don’t have it or can’t see it, look up how to enable it in your version of Excel.

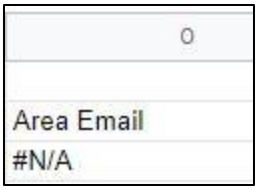
11. Click “Macros”.



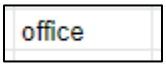
12. Look for the macro named “Convert_Vehicles_Assignment_Report_For_VMLC”. If you do not have this macro, you will have to import it. (Go to Chapter 3 part 4)

13. Double click on it or hit “Run” while it is highlighted.

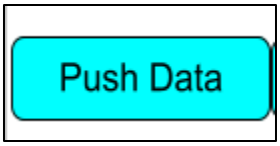
14. It should then format the report for the “Master Data” sheet in the “Virtual Mileage Logs Controller (VMLC)”.
15. Next you will need to copy the finished report into the “Master Data” sheet.
16. Hit CTRL + A (select all)
17. Hit CTRL + C (copy)
18. Open the “Master Data” sheet.
19. Click on cell “A1”
20. Hit CTRL + V (paste)
21. Check the O and P columns for errors in the “Master Data” sheet.



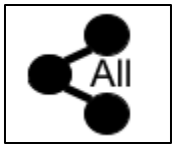
Errors are present when the area in column C is not present in the "List of all area emails" sheet. This means one of two things. Either the "List of all area emails" sheet has not been updated or the car is a mission vehicle. If it is because the car is a mission vehicle change the area in column C to "office".



22. Run the "Push Data" function then the Share with ALL function.



THEN



Note: Alternatively, you can manually update the "Master Data" sheet and imos separately if you so choose. Just update cell E3 in the "Script Master Sheet" and then run the "push data for 1" button then the "share One" button per car you update.

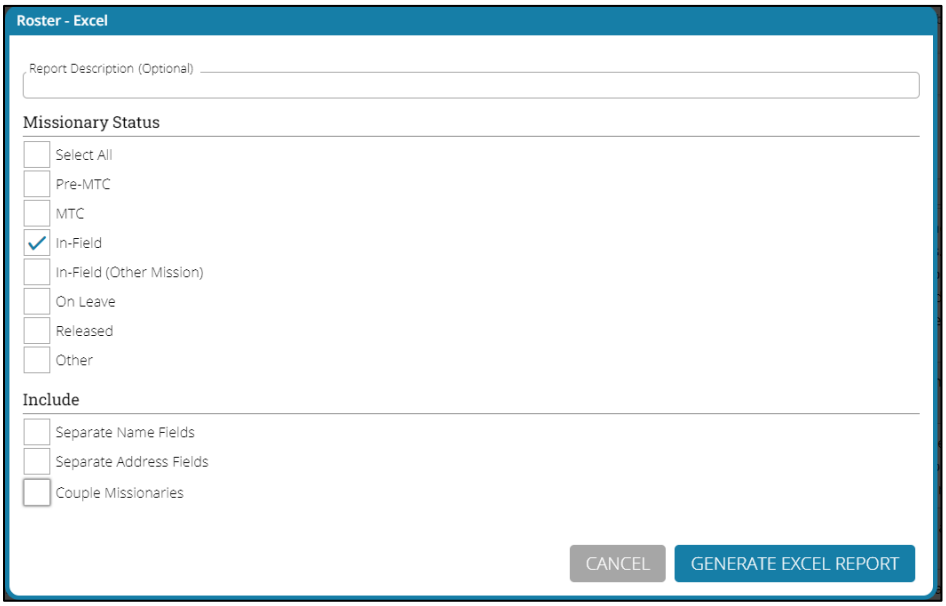
Part 2 – Updating "List of all area emails"

To keep "Virtual Mileage Logs Controller (VMLC)" up to date and working properly you will need to constantly update information in the "List of all area emails" sheet. This sheet contains a list of the area emails. It is used so that each log has an associated area email. Its format is a variation of the "Roster - excel" report pulled from IMOS. The steps to update the "List of all area emails" are as follows:

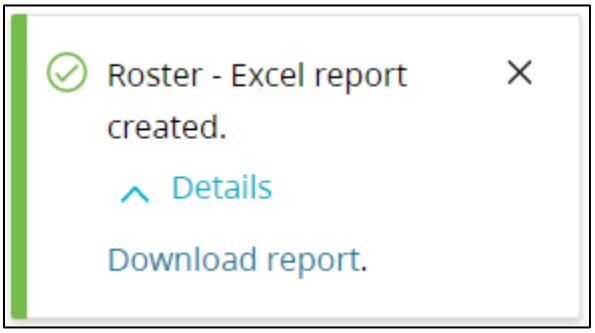
1. Log on to IMOS/the Missionary Portal
2. Navigate on the left-hand side panel and look for "Reports and Surveys"

3. Click on "Mission Office Reports"
4. Click on the "Roster - Excel" report.

5. When you download the “Roster - Excel” report it will ask you a few things, make the settings as follows when downloading this report. The only box that should be checked is the “In-Field” checkbox.



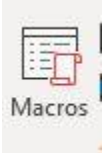
6. Download the “Roster - Excel” report.



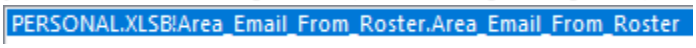
7. Open the “Roster - Excel” report in Excel.
8. Turn off “Protected View”

9. Switch to the Developer Tab. If you don’t have it or can’t see it, look up how to enable it in your version of Excel.

10. Click “Macros”.



11. Look for the macro named “Area_From_Roster”. If you do not have this macro, you will have to import it. (Go to Chapter 3 part 4)



12. Double click on it or hit “Run” while it is highlighted.

13. It should then format the report for the “List of all area emails” sheet in the “Virtual Mileage Logs Controller (VMLC)”.
14. Next you will need to copy the finished report into the “List of all area emails” sheet.
15. Select all the data in columns A – B.
16. Hit CTRL + C (copy)
17. Open the “List of all area emails” sheet.
18. Click on cell “A1”
19. Hit CTRL + V (paste)

Part 3 – Updating Cars mid-Month and mid-Transfer

Sometimes you may not want to update the entire “Master Data” sheet or the “List of all area emails” sheet all at once. The option to update these manually is still an option. Just be sure to update everything you change in these sheets in IMOS as well. If you are only updating a single car, it is better to do it manually. Every time you run the “Share with ALL” button/function it sends an email to every area email involved. So, it is better to not spam area emails with emails they could go without. The process to update a single car log’s information is as follows:

Change everything in IMOS first.

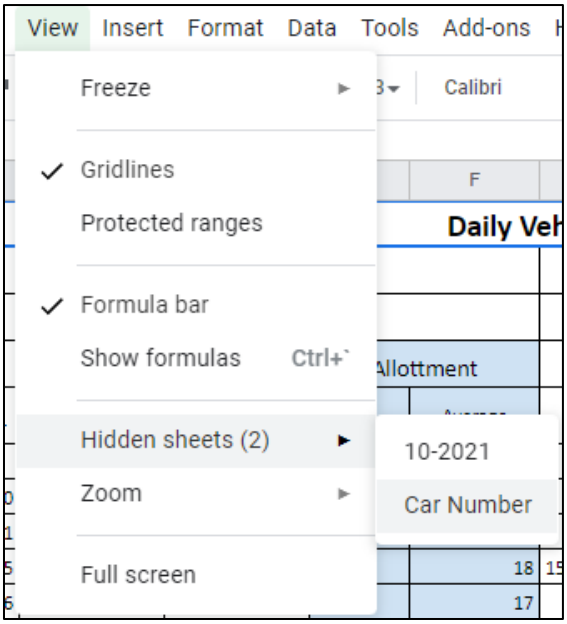
Navigate to the “Master Data” sheet and find the car you will be modifying. (*If you are doing a car switch it is recommended you use the “Switch Cars” function/button – See in chapter 5.*)

If the **car is being reassigned** to a different area, update the Zone, District, and Area columns respectively. The Area Email column is dependent on the area column and will update automatically.

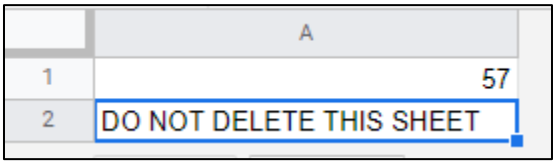
Zone	District	Area	Area 2

If the **car number needs to change** the process is a little more complicated. There are two things you will have to change to change the car number.

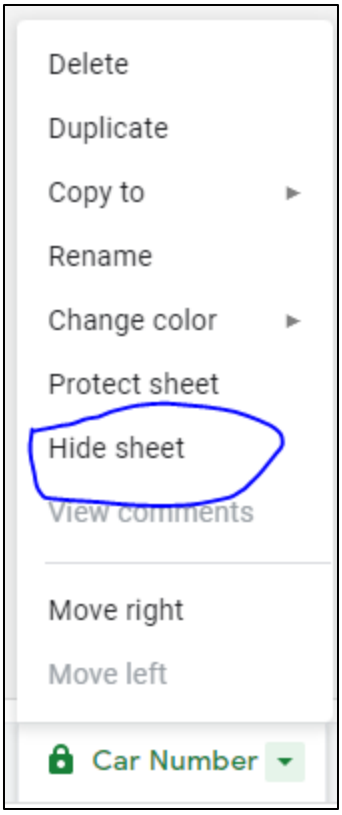
1. Change the car number in the “Master Data” sheet. If this is a new car, create a new column, DO NOT CHANGE EXIXSTING DATA FOR NEW CARS.
2. You will need to change the car number in the actual car log for this car.
3. Open the changed car number car log.
4. Click on “View”
5. Click on “Hidden Sheets”
6. Click on “Car Number”



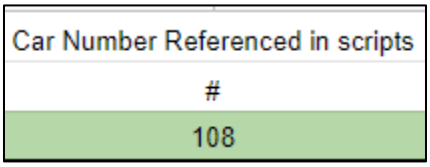
7. Inside the “Car Number” sheet should be two cells.



8. Cell “A1” contains the car number for this log. If this value is changed the car log will be identified as the new value. **WARNING! IF THIS NUMBER IS INCORRECT, SCRIPTS WILL ALSO REFRENCE THIS CAR LOG INCORRECTLY!**
9. Change cell “A1” to the new car number. It is also recommended to change the car number referenced in every month’s sheet (but not required – to prevent confusion).
10. After changing the car number, hide the “Car Number” sheet. This is done by right clicking the “Car Number” tab at the bottom of the screen then clicking “Hide sheet”.



11. Once the car number has been updated, update cell E3 on the "Script Master Sheet" to reflect the (new) car number you just updated. (If I updated car 1 to be car 2, I would put 2 in cell E3).



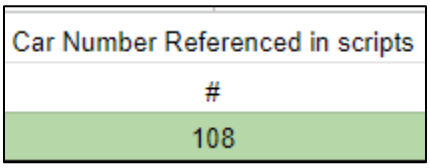
12. Run the "Push Data for 1" function/button and wait for function/script to finish.

13. Run the "Share with One" function/button and wait for function/script to finish.

If you are **Changing the Monthly Allowed Miles**, the process is quite simple.

1. Change the monthly allowed miles for the car you would like.

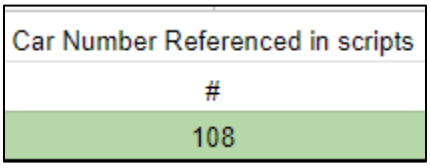
2. Update cell E3 on the "Script Master Sheet" to reflect the car number you updated the miles for.



3. Run the "Push Data for 1" function/button and wait for function/script to finish.

If you are **Changing the Designated Driver/Drivers**, the process is simple.

1. Change the designated driver/drivers for the car you would like.
2. Update cell E3 on the "Script Master Sheet" to reflect the car number you updated the driver for.

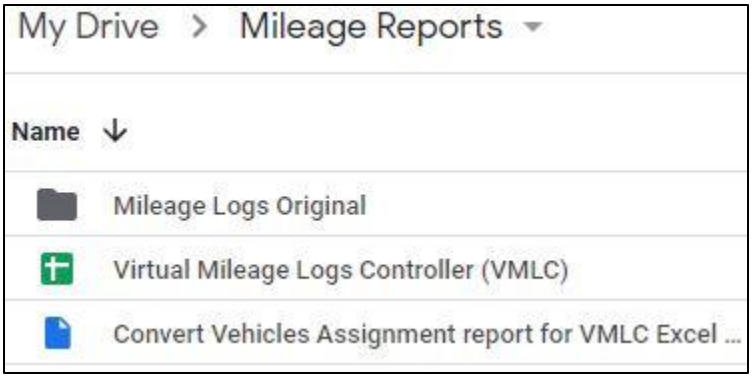


3. Run the "Push Data for 1" function/button and wait for function/script to finish.

Part 4 – How to import Visual Basic macros into Excel

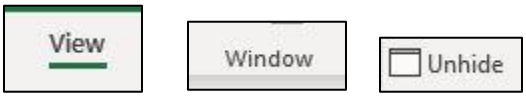
This section may be out of date by the time you read it, never fear for a quick Google search is here! If the instructions in this section do not work for you, do a quick Google search for "How to import .bas file into personal workbook Excel".

1. The file you are looking for should be in the same folder in your Google Drive as the "Virtual Mileage Logs Controller (VMLC)" file. If either of the files are not in the same place as the "Virtual Mileage Logs Controller (VMLC)" file, then they are available here:
   Area_Email_From_Roster.bas
   Convert Vehicles Assignment report for VMLC.bas
   Download this file.

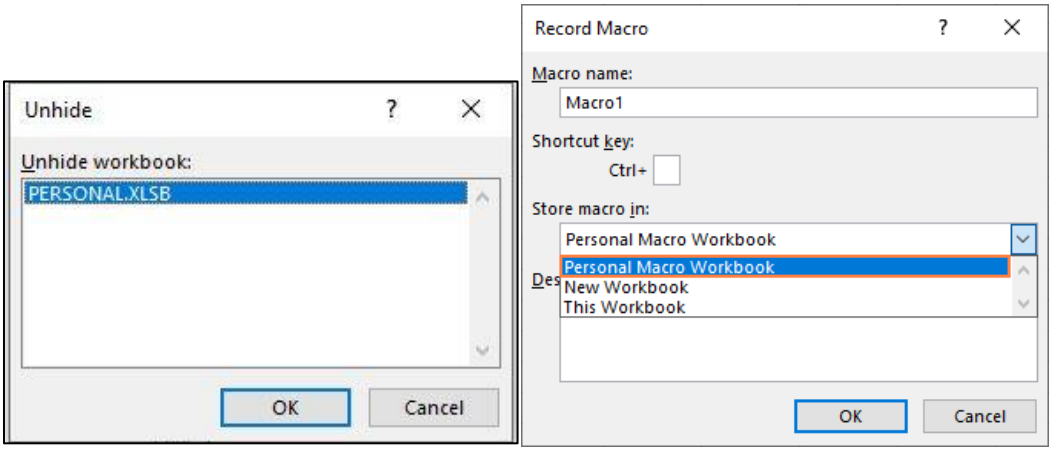


Once the file is downloaded you will want to import it to your personal workbook in Excel. To import it into your personal workbook you will first have to unhide your personal workbook.

2. Open Excel
3. Click on the "View" tab. Then in the "Window" category, click on the "Unhide" button.



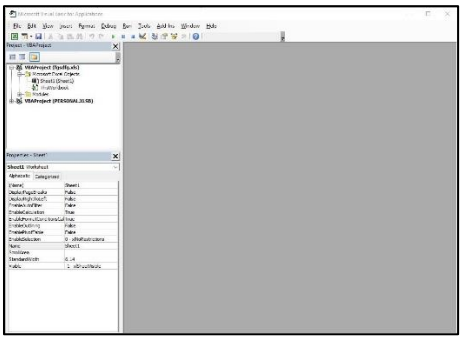
4. A dialog will popup asking you what spreadsheet you would like to unhide. Click the "PERSONAL.XLSB" option, then click "OK". If you do not see the "PERSONAL.XLSB" option, you need to create the "PERSONAL.XLSB" file. You can do this by recording a macro and saving it in your personal workbook. If you are still having trouble, search online for "how to show personal workbook excel" for your version of excel.



5. Once the personal workbook is unhid, switch to the Developer Tab in the newly opened spreadsheet.

6. Click "Visual Basic".

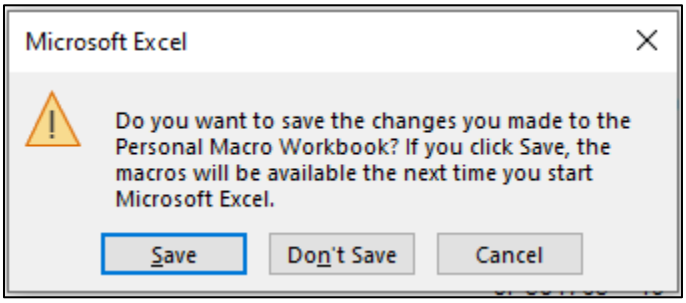
7. A window named “Microsoft Virtual Basic for Applications” will then open. It looks something like this.



8. Click “File”, then click “Import File”

9. A file explorer window will then open. Find the .bac file you just downloaded in step 1.
10. Open that file.
11. Close the Virtual Basic window.
12. In excel, click on the “View” tab. Then in the “Window” category, click on the “Hide” button.

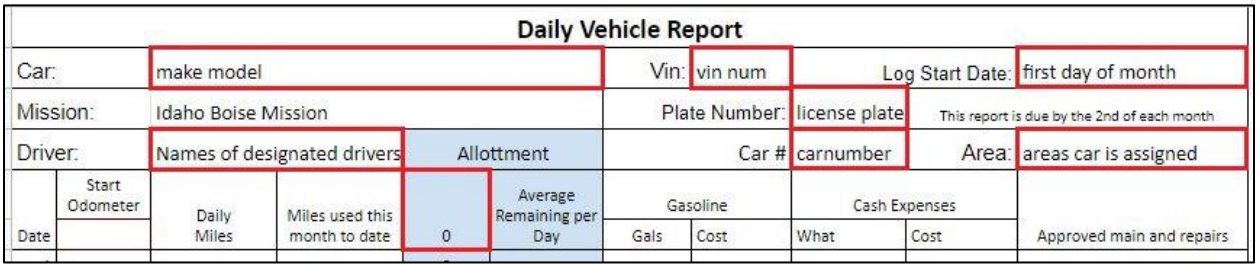
13. Close all Excel windows except for the ones prior to starting these steps. If you are asked to save the changes made in your personal workbook, click “Save all” or “Save”



Chapter 4 - Automated Actions Done for You

Part 1 – Creating Next Month's sheet

Every first of the month from 2-3am a script/function is run that creates the next month's sheet. If the new month is November 2021, the script will create a new sheet named "11-2021" in every present mileage log. Each new month is created based on the "Log Template" present in the "VMLC" spreadsheet. If the sheet ("11-2021") in the mileage log already exists, it pushes the most updated information to this sheet and does not use the "Log Template" (it assumes that it has already been created). This function automatically knows what the next sheet is named. This function will not delete already "put-in" odometer entrees. It will only update the car information on the newly created sheet: such as designated driver, car number, etc. See picture. Updated fields are surrounded by red.



Daily Vehicle Report											
Car:	make model				Vin:	vin num		Log Start Date:	first day of month		
Mission:	Idaho Boise Mission				Plate Number:	license plate	This report is due by the 2nd of each month				
Driver:	Names of designated drivers		Allottment		Car #	carnumber		Area:	areas car is assigned		
	Start Odometer					Gasoline		Cash Expenses			
Date		Daily Miles	Miles used this month to date	0	Average Remaining per Day	Gals	Cost	What	Cost	Approved main and repairs	

If the function is ever to malfunction or stop working entirely (due to code depreciation) you will need to disable/delete it's run trigger (what makes it run every month). To disable/delete the trigger go to chapter 4 part 4 – How to disable/delete a trigger. You will be looking for "triggerCreateNextMonth".

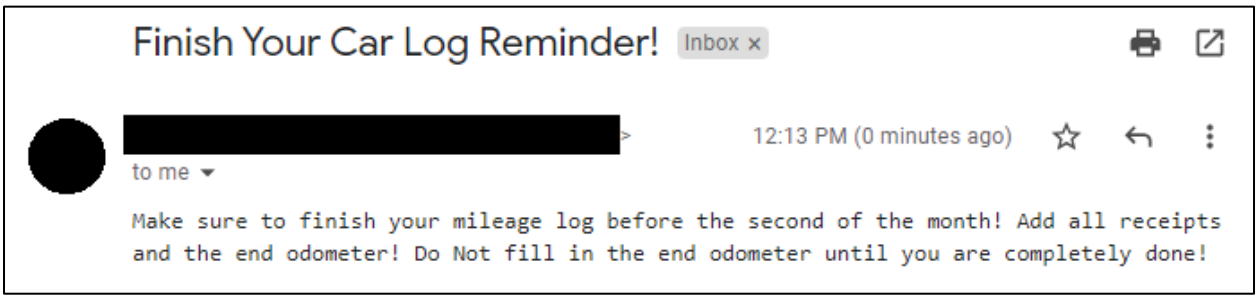
Part 2 – Email reports to you on the 3rd of the month

On the 3rd of the month at 2am-3am, a script/function is run that will send you a landscape pdf report for each car log present. The pdf report will be for the month that just completed. For example, if the date is November 3, 2021, the report emailed to you will be for October 2021 (sheet "10-2021"). This function is also available as a user available script.

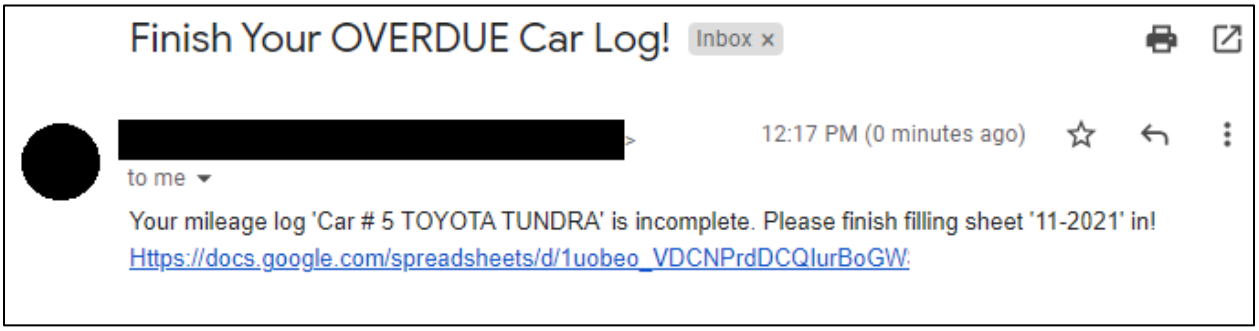
If the function is ever to malfunction or stop working entirely (due to code depreciation) you will need to disable/delete it's run trigger (what makes it run every month). To disable/delete the trigger go to chapter 4 part 4 – How to disable/delete a trigger. You will be looking for "triggeremailAllReportsLandscape".

Part 3 – Missionary Email reminders on 1st and 3rd of the month

Every month missionaries will get a maximum of two reminder emails to finish their mileage logs. The condition that these emails are sent is if whether or not the mileage logs are complete (end odometer filled in). The first email will be sent on the 1St of the month between 5am and 6am reminding those who have not completed their mileage logs to finish them. This email will also inform them when the mileage logs are due.



Then on the 3rd of the month between 5am and 6am they will receive another email if they still have not completed their mileage logs.



If the function is ever to malfunction or stop working entirely (due to code depreciation) you will need to disable/delete it's run trigger (what makes it run every month). To disable/delete the trigger go to chapter 4 part 4 – How to disable/delete a trigger. You will be looking for is "triggerfinishMileageLogsReminderEmail" and "triggerincompleteMileageLogsEmail".

Part 4 – Completed sheets are protected from editing

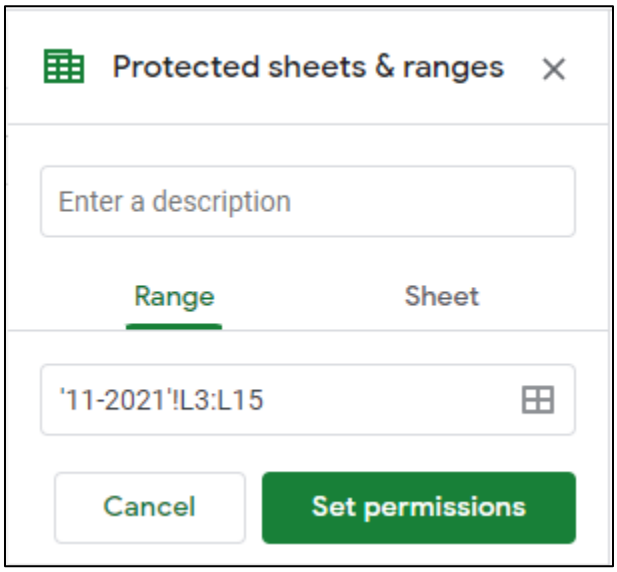
This function protects all the sheets (in each mileage log) from editing except those that are incomplete (end odometer filled in). This lowers the risk that previous months will be modified/tampered with. This function automatically runs on the 3rd of the month between 12 – 1 AM and the 6th of the month between 12 – 1 AM. As a disclaimer for pervious months sheet - if missionaries have filled in their end odometer but have not

actually finished their milage log (missing receipts or odometer entrees). When this function is run, they will lose edit access to the sheet. If this is to happen follow the following steps:

1. Open the affected mileage log.
2. Navigate to the “Data” tab in the file.

3. Click “Protect sheets and ranges”.

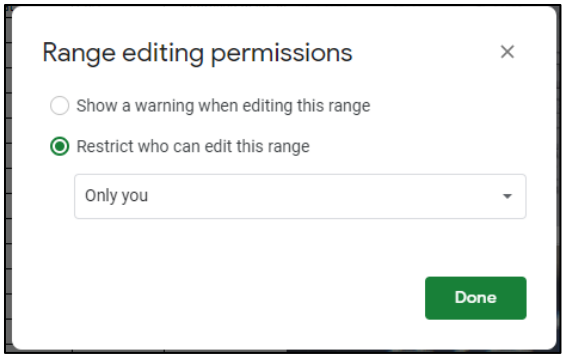
4. It’s then going to try to protect the current selection of cells. We don’t want that. Press “Cancel”.



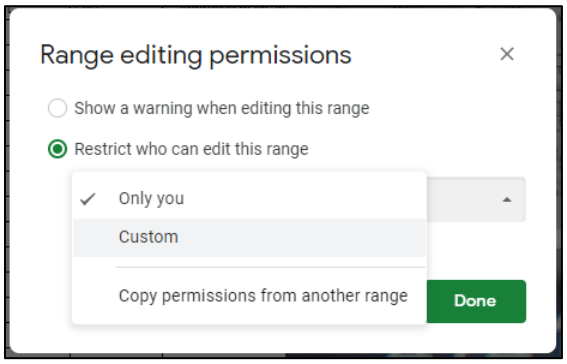
5. Then you need to be sure that you are looking at all the protected ranges present in the google spreadsheet. To do this click on “Show all protected ranges”.

6. You then will be able to see all the protected ranges. Look for the previous months sheet (the one the missionaries can’t edit). Click on it.
7. Then click “Change permissions”

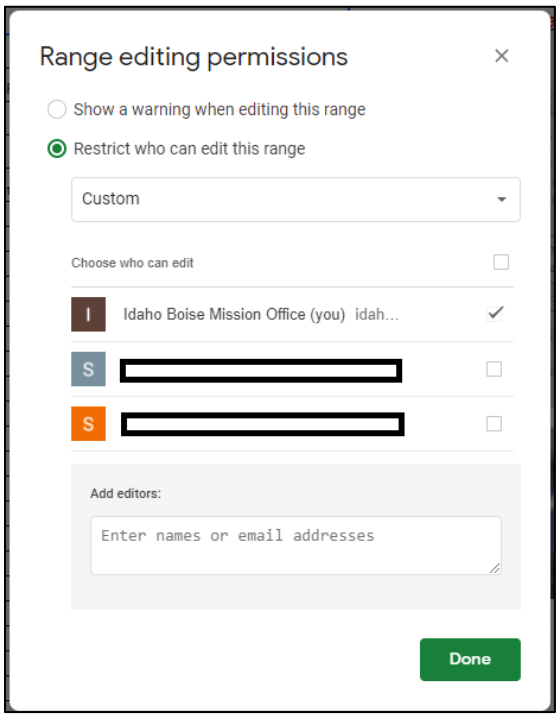
8. The default “Range editing permissions” dialog looks like this.



9. Click on “Only you”.
10. Press “Custom”



11. A new dialog will show up looking something like this.



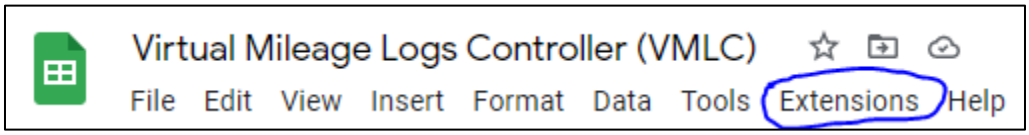
12. Then just check the area emails present (only ones assigned to this log should be there).

If the function is ever to malfunction or stop working entirely (due to code depreciation) you will need to disable/delete it's run trigger (what makes it run every month). To disable/delete the trigger go to chapter 4 part 4 – How to disable/delete a trigger. You will be looking for the two "triggerprotectSheets" triggers.

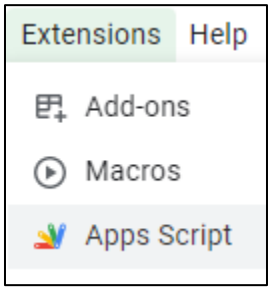
Part 5 – How to disable/delete a trigger

Open the VMLC spreadsheet in google sheets.

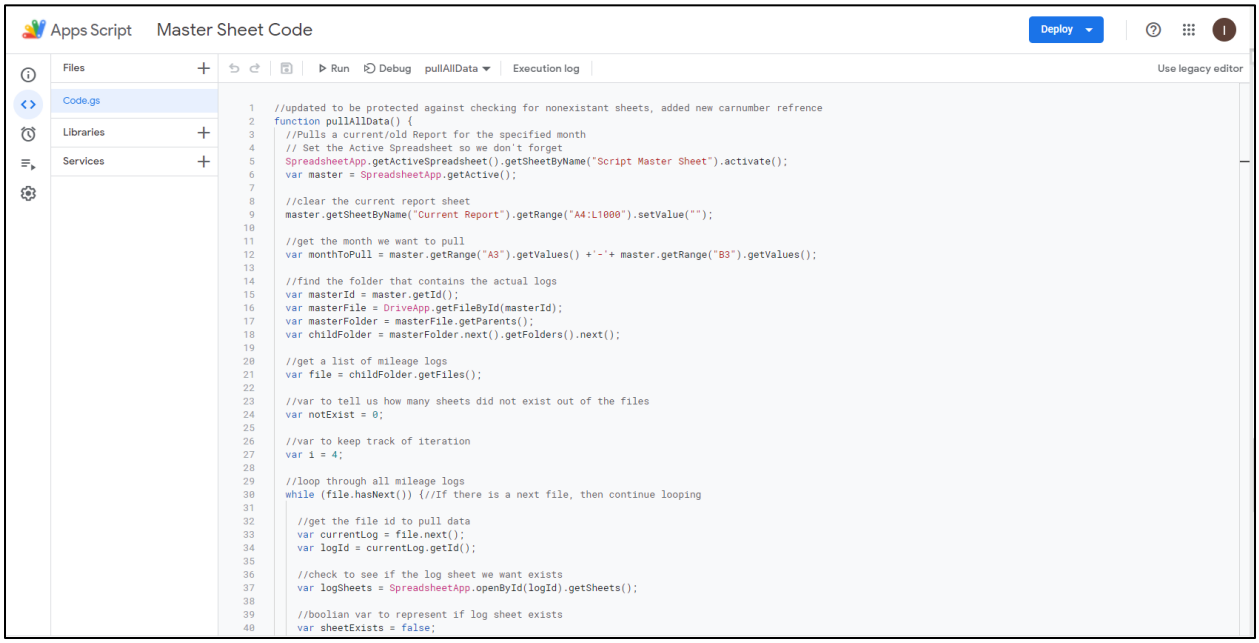
1. Click on "Extensions"



2. Click on "Apps Script"

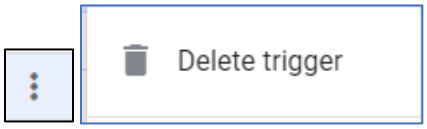


3. A new tab will open showing you the google apps script code. It looks something like this.



4. Click on the alarm icon on the left panel of the screen (triggers).

5. When this project was in its initial glory there were 6 triggers which are all covered in this chapter (chapter 4). Look for the trigger you were planning on disabling/deleting.
6. Once you have located the trigger, click on the three dots button then hit “Delete trigger”.



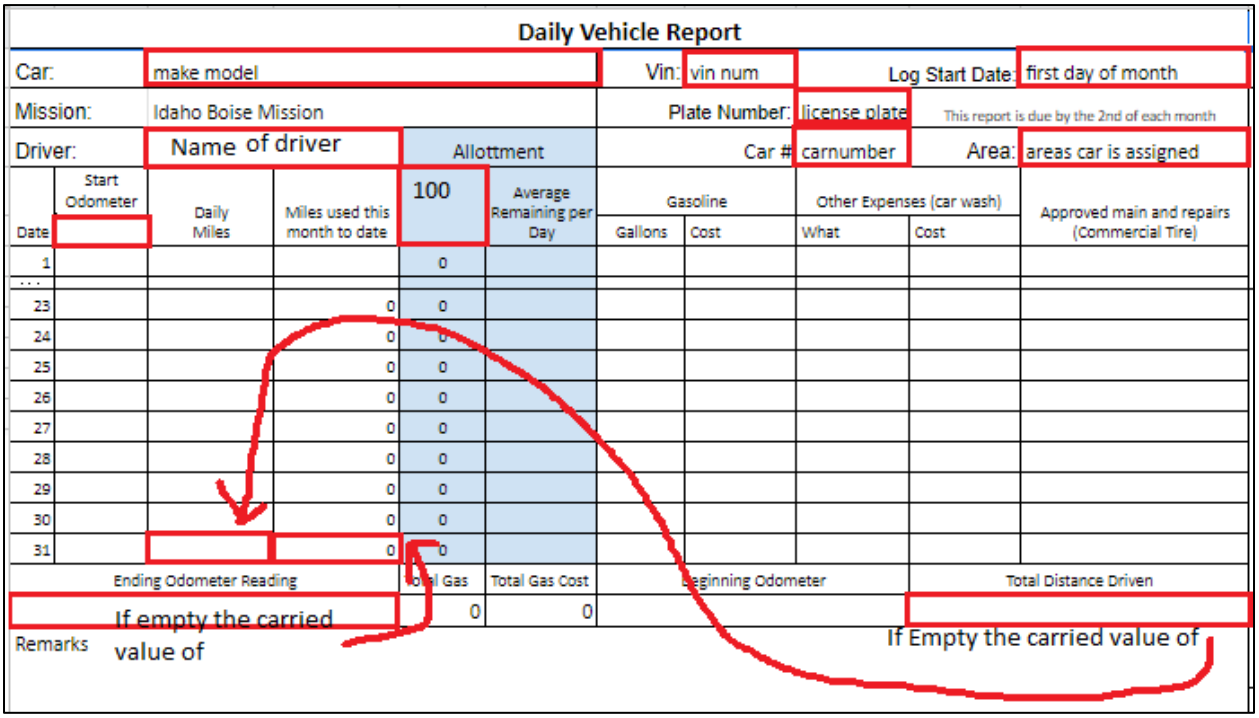
Chapter 5 - User Available Scripts

Part 1 – Pull a current report

This function pulls information from each milage log to provide you with current information.

Pull a current report (script)	Description:
Current Report	Pulls a current report from all the vehicle logs from the sheet '11-2021'. It pulls the information to the 'Current Report' sheet. Change cells A3 and B3 to change the month the report is for. The current report does not update automaticlly, only when the 'Pull data' button is pressed. (Script is dependant on cells A3 and B3)

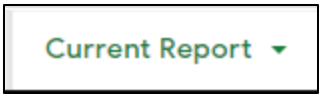
This pulled/current information includes: The car number, The number of miles driven, the miles allowed, when the log started, the area(s) assigned to the car, the start odometer, the end odometer, the designated driver(s), the make and model, the license plate number, the vin number (last 8), and the file URL (for convenience). Picture below for visual representation.



Daily Vehicle Report

Car:	make model				Vin:	vin num		Log Start Date:	first day of month	
Mission:	Idaho Boise Mission				Plate Number:	license plate		This report is due by the 2nd of each month		
Driver:	Name of driver		Allottment		Car #:	carnumber		Area:	areas car is assigned	
Date	Start Odometer	Daily Miles	Miles used this month to date	100	Average Remaining per Day	Gasoline Gallons	Gasoline Cost	Other Expenses (car wash) What	Other Expenses (car wash) Cost	Approved main and repairs (Commercial Tire)
1				0						
...										
23			0	0						
24			0	0						
25			0	0						
26			0	0						
27			0	0						
28			0	0						
29			0	0						
30			0	0						
31			0	0						
Ending Odometer Reading				Total Gas	Total Gas Cost	Beginning Odometer		Total Distance Driven		
If empty the carried value of				0	0			If Empty the carried value of		
Remarks										

Once the information is pulled from each mileage log, that information will be shown in the "Current Report" sheet in the "Virtual Mileage Logs Controller (VMLC)" spreadsheet.



Keep in mind the Cell Color Key that help you find issues with the mileage logs.

Cell Color Key	Over Miles	Miles Driven at or below 0
	Begining Odomiter empty	Driver Log not complete

Here is a sample Current Report.

Current Report				Cell Color Key	Over Miles	Miles Driven at or below 0					
					Begining Odomiter empty	Driver Log not complete					
Car	Miles Driven	Miles Allowed	Log Start	Area	Start Odomiter	End Odomiter	Designated Driver	Make and Model	Licence Plate	Vin	File URL
5	0	999	11/1/2021	office		Driver log incomplete currently ~ 0	Fred Figglehorn	TOYOTA TUNDRA	39ridkdfgt	65564cf5	https://docs.g
5	10000	999	11/1/2021	office	100	10100	Elder Ransom	Ford Car	298uy92	76999NH7	https://docs.g

DEPENDANCIES:

This function is dependent the “Month and Year Referenced in Scripts” so be sure it reflects the month and year you are wanting to pull a report for.

Month and Year Referenced in Scripts	
Month (##)	Year (####)
11	2021

Part 2 – Push Data/Push Data 1

This function pushes data from the “Master Data” sheet to the mileage logs.

Push Data to '11-2021' (script)	Description:
Push Data / Push Data for 1	Updates the information on sheet '11-2021' to match what is shown in 'Master Data'. This includes driver's name, area, etc. 'Push Data' will update the information for all the driver logs, 'Push Data 1' will update the information for car # 5. (Script is dependant on cells A3, B3, and the entire 'Master Data' sheet.)

To get a better grasp on what this function does exactly in the mileage log see this image. The fields surrounded by red are those fields that are updated with information from the “Master Data” sheet.

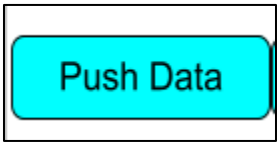
Daily Vehicle Report											
Car:	make model					Vin:	vin num	Log Start Date:	first day of month		
Mission:	Idaho Boise Mission					Plate Number:	license plate	This report is due by the 2nd of each month			
Driver:	Names of designated drivers			Allottment		Car #	carnumber	Area:	areas car is assigned		
Date	Start Odometer	Daily Miles	Miles used this month to date	0	Average Remaining per Day	Gasoline		Cash Expenses		Approved main and repairs	
						Gals	Cost	What	Cost		

The Differences between "Push Data" and "Push Data for 1"

"Push Data for 1" does everything this section talks about for one mileage log. What mileage log is this? It is the one specified in the "Car Number Referenced in scripts" section.

Car Number Referenced in scripts
#
101

"Push Data" is much like "Push Data for 1". "Push Data" does the same thing as "Push Data for 1" except for all the mileage logs, not just one. So it is not dependent on the "Car Number Referenced in scripts" section.



DEPENDANCIES:

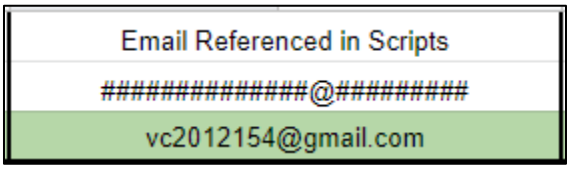
This function is dependent the "Month and Year Referenced in Scripts" so be sure it reflects the sheet you are wanting to push information to. (Image below would interact with sheet "11-2021".) The "Push Data for 1" is dependent on the "Car Number Referenced in scripts". It is also dependent on the on information in the "Master Data" sheet.

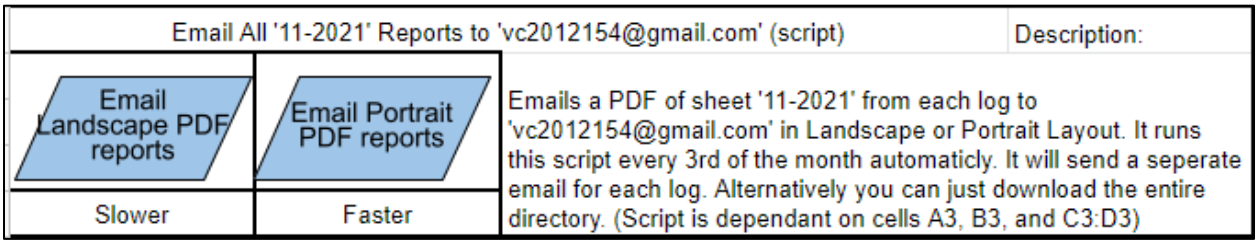
Month and Year Referenced in Scripts	
Month (##)	Year (####)
11	2021

Car Number Referenced in scripts
#
101

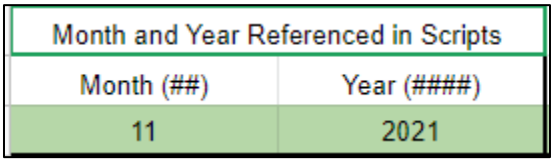
Part 3 – Email PDF Report Landscape/Portrait

This will email a PDF report of each mileage log to the email declared in the "Email Referenced in Scripts" section. One email per log will be sent.





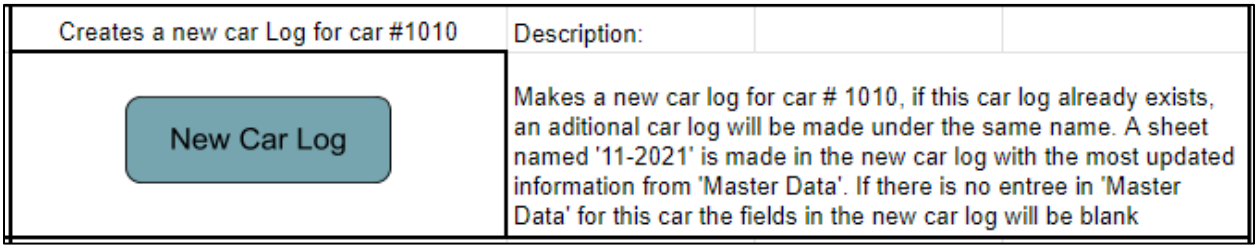
The PDF emailed will be of the specified month in the "Month and Year Referenced in Scripts" section. So for the image below, the sheet emailed to you as a PDF would be "11-2021".



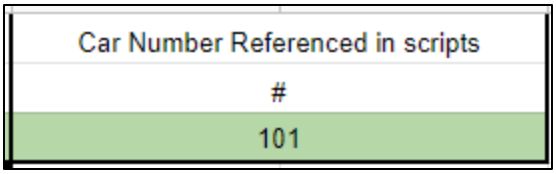
DEPENDANCIES:

This function is dependent the "Month and Year Referenced in Scripts" section so be sure it reflects the month you are wanting PDFs for. This function is also dependent on the "Email Referenced in Scripts" section.

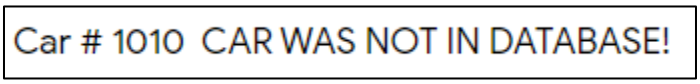
Part 4 – New Car Log



This function creates a new car log. The car log that is created is filled with the information associated with the car declared in the "Car Number Referenced in scripts" section.



If the information is not present in the “Master Data” sheet for the declared car, the created car log will be similar to the “Log Template” sheet and the file name will reflect that the car was not found.

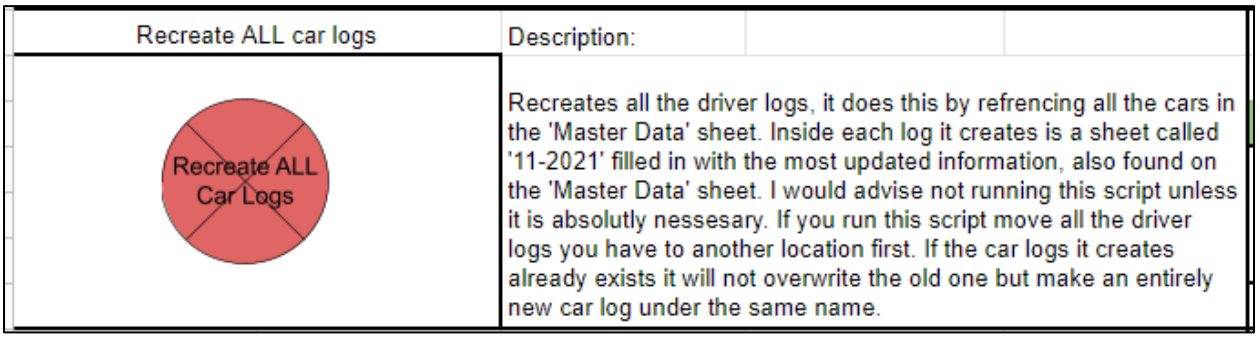


This function doesn’t care if the file already exists, it will always create a new log and will never overwrite another log.

DEPENDANCIES:

This function is dependent the “Month and Year Referenced in Scripts” section so be sure it reflects the current month and year. This function is dependent on the “Log Template” sheet. This function is dependent on information being present in the “Master Data” sheet, be sure there is information for the specific car you are making before creating a new car log.

Part 5 – Recreate ALL Car Logs



This function is much like the “New Car Log” function. This function creates a car log for every car present in the “Master Data” sheet. In each log it pushes the most current information present in the “Master Data” sheet. The information that is updated is shown below.

Daily Vehicle Report											
Car:	make model					Vin:	vin num	Log Start Date:	first day of month		
Mission:	Idaho Boise Mission					Plate Number:	license plate	This report is due by the 2nd of each month			
Driver:	Names of designated drivers			Allottment		Car #	carnumber	Area:	areas car is assigned		
Date	Start Odometer	Daily Miles	Miles used this month to date	0	Average Remaining per Day	Gasoline Gals	Gasoline Cost	Cash Expenses What	Cash Expenses Cost	Approved main and repairs	

This function doesn't care if the file/driver log already exists, it will always create a new log and will never overwrite another log. Generally this function would be used once every year to prevent clutter in each mileage log, by cycling out old mileage logs and replacing with new ones. (Mileage logs with many months in them (old) to mileage logs with few months in them (new).)

DEPENDANCIES:

This function is dependent the "Month and Year Referenced in Scripts" section so be sure it reflects the current month and year. This function is dependent on the "Log Template" sheet. This function is dependent on information being present in the "Master Data" sheet, it will only create logs for cars with information.

Part 6 – Share with Proper Area Emails ALL/One

Share With Proper Area Emails	Description:
Share Car # 1010 with it's area email/s All One	Shares each car log with the proper area email declared in 'Master Data'. It also removes all previously added editors, so only the ones actually needing access are given that access. Press 'All' to share all the logs with the proper area email address respectivly. Press 'One' to share Car Log # 1010 with it's proper area email address.

This function shares each/one of the milage logs with its assigned area email. The assigned area email can be found in the "Master Data" sheet. This function is used if the car assignment changes. (For example, if car #1 is assigned to area RV 2, then the car is switched to area RV 3 because of a medical reason, this function would be used to then share car #1 to RV 3 and remove driver log access from RV 2.) **Warning!** Be conservative in using either of these functions, as for every log that is shared/reshared, an email will be sent to each corresponding area email. (RV 3 would be sent an email.)

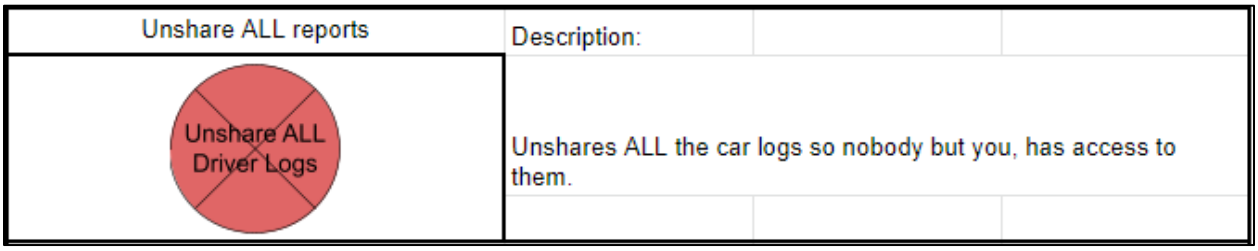
The differences between "Share All" and "Share One".

"Share One" will share the mileage log associated with "Car Number Referenced in scripts" with its associated area email. "Share All" will share each mileage log with its associated area email. (The area email is declared in the "Master Data" sheet.)

DEPENDANCIES:

This function is dependent on information being present in the "Master Data" sheet, so be sure the declared email is correct for each car. "Share One" is dependent on the "Car Number Referenced in scripts" section.

Part 7 – Unshare ALL Reports

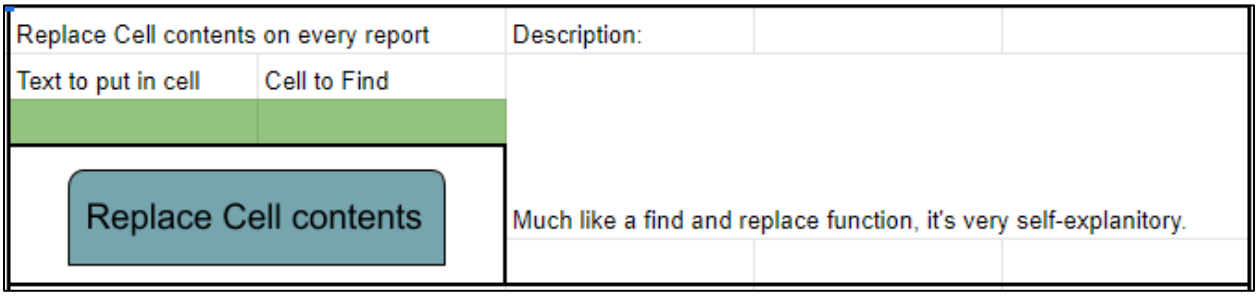


This function is an emergency only script. Meaning it shouldn't ever need to be used. It will Unshare ALL the driver logs present so nobody except the owner of said logs has any type of access to them (you are the owner).

DEPENDANCIES:

None.

Part 8 – Replace Cell Contents on Every Report



This function is much like a find and replace dialog. In the "Cell to Find" enter a cell location such as "C5" then in the "Text to put in cell" enter the text you would like in that

found cell. For example if you wanted every C5 cell to say “Yes” in every milage log for the sheet “11-2021” you would make sure that “Month and Year Referenced in Scripts” reflects 11 and 2021. Then you would enter C5 in the green cell under “Cell to Find” and then in the green cell below “Text to put in cell” you would enter Yes.

DEPENDANCIES:

This function is dependent the “Month and Year Referenced in Scripts” so be sure it reflects the month you are wanting to find and replace text/cells on.

Month and Year Referenced in Scripts	
Month (##)	Year (####)
11	2021

Part 9 – Check Scripts/Logs for Errors

Check Scripts/Logs for Errors	Descripton:
Search Logs For Errors	Checkes for the most common mistakes / problems that could of been created that cause errors when running scripts. If it finds the problem it should suggest to you how to fix it.

If every you are to run into errors in your day to day use of this project, you may want to run the “Search Logs For Errors” script/function to see if the type of error you are running into is fixable by you. The script does not check for everything that could possibly go wrong. Generally if you run into a problem, it is a simple fix or caused by some small mistake. For the most part you should not have to run this script. Also keep in mind that server errors are temporary problems so if you run into one just run your script again.

DEPENDANCIES:

None. It finds fault in other scripts dependencies.

Part 10 – Switch Cars

Switch Cars		Description:
Car 1	Car 2	Switches the information assosiated with each car in the 'Master Data' sheet, pushes that updated information to it's respective driver logs, shares it with the correct email address, and protects unuthorized editing for the newly shared emails.
1	2	
Switch Cars		

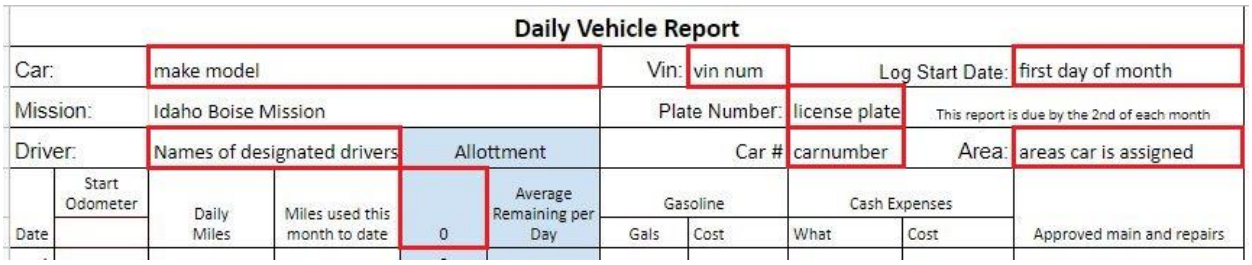
This function is very useful in mid-transfer or mid-month car switches. Like its description depicts, this function switches the information for two cars in the "Master Data" sheet and then pushes the newly updated information to the car logs. Once it is done doing that it shares the modified milage logs with their respective, newly assigned, area email. For a visual representation of this process see the image sequence below.

This is a modifed Vehicle assignments rep Info is pulled fr

Zone	District	Area	Area 2	Vire	Monthly Allowed	Units	Designated Driver	Designated Driver 2	Area Email	Area Email 2
zone 5	district e	ccc		rar	875	Miles	Joe		colby.ransom@missionary.org	
zone 2	district v	cccc		rar	1800	Miles	Tarnish		colby.ransom@missionary.org	

UPDATES AUTOMATICLY

The information Switched in the "Master Data" sheet.



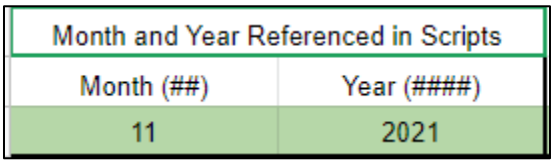
Daily Vehicle Report

Car:	make model			Vin:	vin num		Log Start Date:	first day of month		
Mission:	Idaho Boise Mission			Plate Number:	license plate		This report is due by the 2nd of each month			
Driver:	Names of designated drivers		Allottment	Car #	carnumber		Area:	areas car is assigned		
Date	Start Odometer	Daily Miles	Miles used this month to date	0	Average Remaining per Day	Gasoline Gals	Gasoline Cost	Cash Expenses What	Cash Expenses Cost	Approved main and repairs

The information updated in each of the two mileage logs following the switch in the "Master Data" sheet.

DEPENDANCIES:

This function is dependent the "Month and Year Referenced in Scripts" so be sure it reflects the current month and year.



Month and Year Referenced in Scripts	
Month (##)	Year (####)
11	2021

This function is dependent on the information in the "Master Data" sheet.

Chapter 6 - Dealing with deprecated or obsolete code.

Part 1 – Deciding what to do

So at some point during the use of the Virtual Mileage Log Controller (VMLC) you may receive an error stating that a certain code sequence is no longer supported. You will then have to decide fully or partially on the future course of the mileage logs. The options I pose at this time, is to either manually maintain the virtual driver logs (with partial or no button support), transition back to physical mileage logs, or move to an in-the-future virtual mileage log system (either provided by the church or the revision of the current system).

Part 2 – Manually maintaining Driver Logs

Manually maintaining the virtual mileage logs is tedious and not recommended unless you are willing to consistently put in a great deal of time, but it is still an option. If you take this option, the majority of the work that you will do is keeping the mileage logs up to date with proper information (such as updating who the designated driver is, what area has the car etc.) and creating a new sheet every month in each log. If you are familiar with Google Sheets these tasks should be no problem for you. I will now cover the basic tasks you will do for these things, along with additional tasks you may end up doing.

Updating information in virtual driver logs

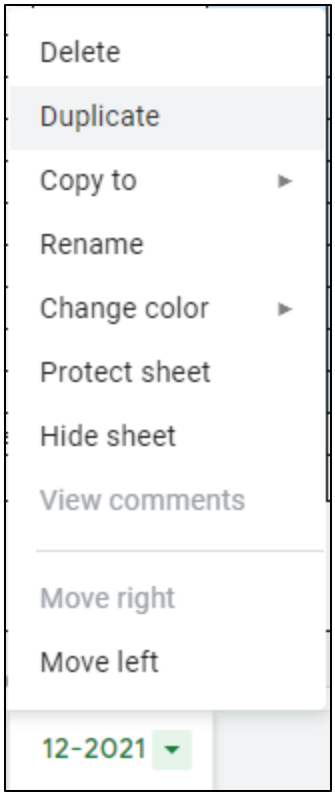
You will need to do this when information for a car change such as when the monthly mile allotment changes or the car is changed to a different companionship in the mission. Many other aspects can change depending on the situation. But the instructions should apply or easily be followed for your specific case.

1. Open the milage log in Google Sheets.
2. Locate the field in the mileage log you want to update.
3. Update the information.

Creating the next month's sheet

This happens every month… pretty self-explanatory. This is done for every mileage log, so keep repeating these steps until all the mileage logs have been updated.

1. Open the milage log in Google Sheets.
2. Right click on the previous month's sheet and press duplicate.



3. It will then create an exact copy of the previous month's sheet. Rename the sheet to match the current month and year.
4. Then proceed to clear the newly created sheet of last month's data. This includes the daily odometer, end odometer, gas receipts, etc.

Sharing the mileage log with the right people

If the companionship driving the car changes you will need to accordingly share the car log with the new companionship and remove the old companionship from the car log. During this process you will need to ensure that previous protections are still in place for the newly added people. (read the next section)

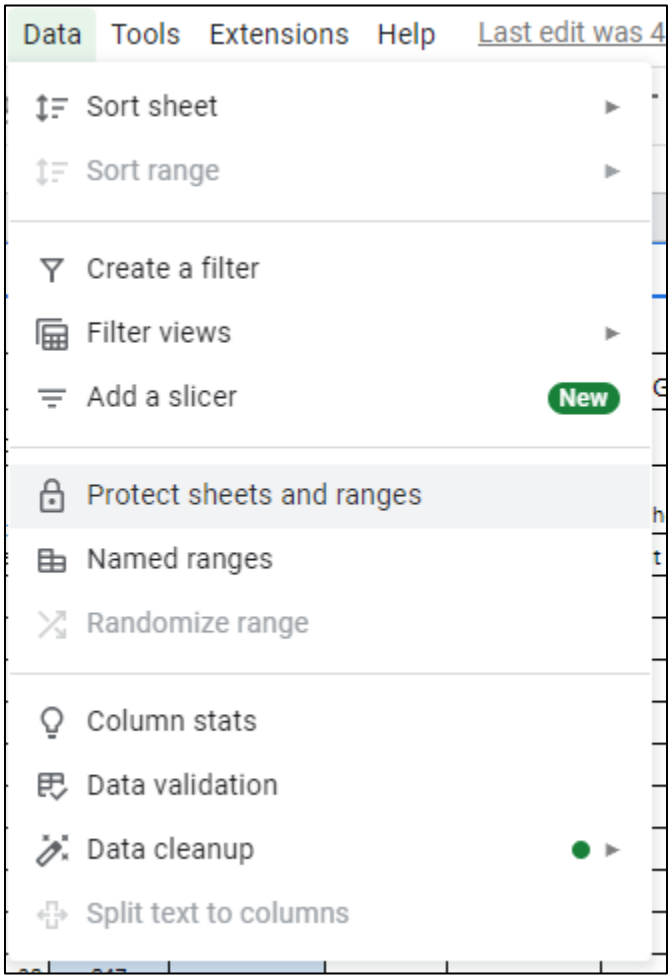
1. Open the milage log in Google Sheets.
2. Click on the "Share" button in the right-hand side of the screen.

3. Remove all accounts listed on this screen except your own.
4. Add the new companionship's area email as an editor. (You can find the companionships email on IMOS in "Teaching Areas".)
5. Press the "Share" button.

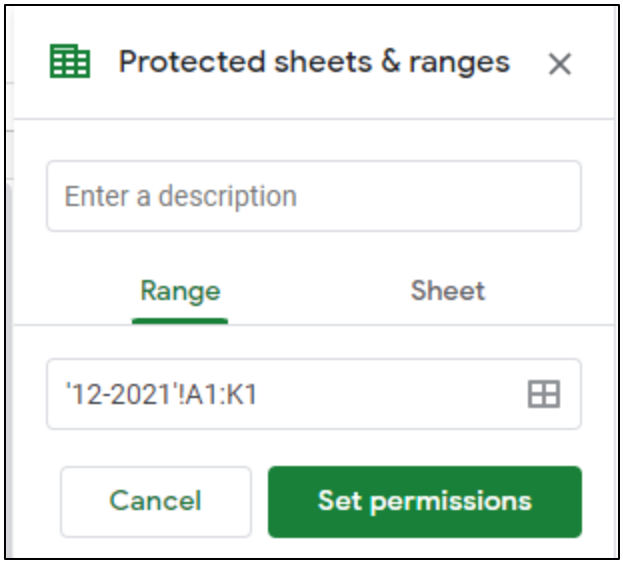
Protecting previous months from unintentional editing

To prevent the accidental or intentional editing of previous months it is recommended to protect previous months sheets. This is done through Google Sheets “Protected sheets & ranges” option.

1. Open the milage log in Google Sheets.
2. Click on “Data” then “Protect sheets and ranges”.

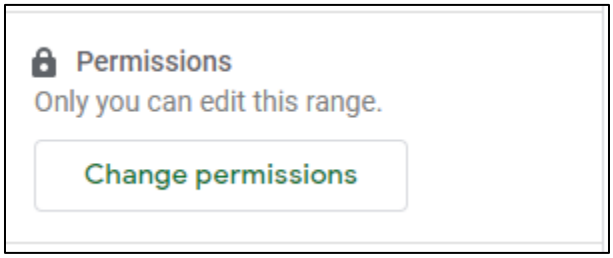


3. If google sheets attempts to protect your current selection, hit cancel.



4. Then click on “Show all protected ranges”.

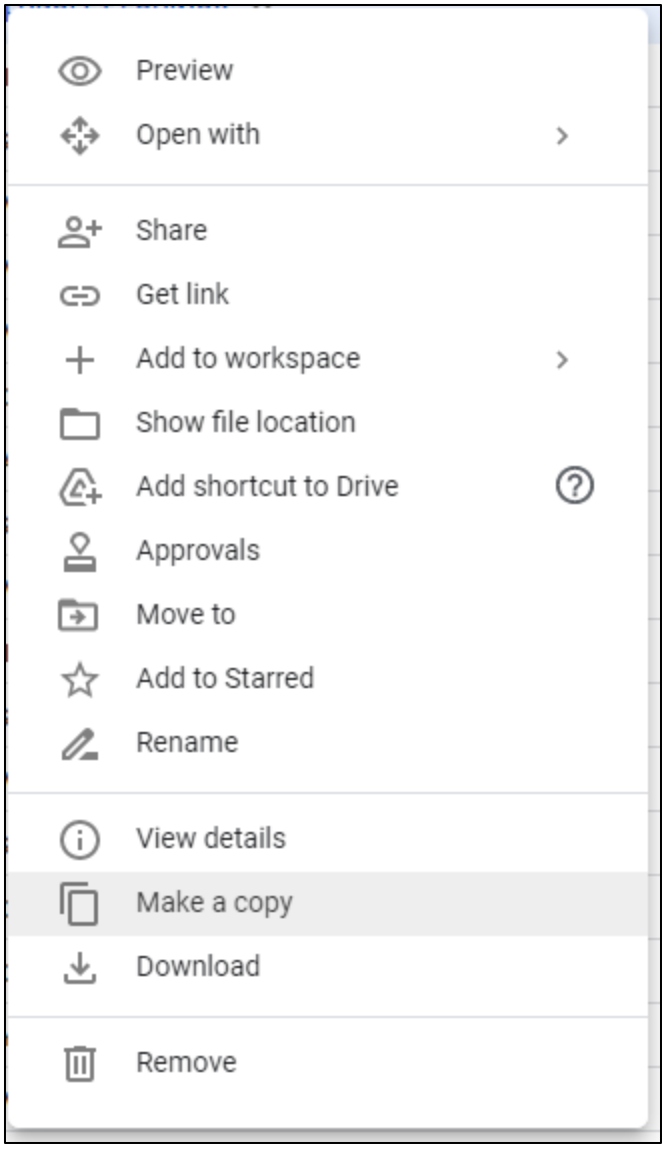
5. Click on every protected range and ensure that the permissions in each protected range are only editable by you. Each protected range should look something like this.



Creating a car log for a new car

If a new car comes into the mission you will eventually need to create a car log for said car. The easiest way to do this effectively is to copy an already in use mileage log and then clear the car specific data from the copy.

1. Locate an in-use mileage log in Google Drive.
2. Right click on the found mileage log and click “Make a Copy”



3. Open the newly copied driver log.
4. Delete all the sheets inside this driver log except for the current months.
5. Then proceed to clear this month's data (the data from the other car). This includes the daily odometer, end odometer, gas receipts, etc.
6. Update the log with the correct information for the new car.

Part 3 – Transitioning to Physical Logs or other.

Transition is always difficult. If it is ultimately decided that the "Virtual Mileage Log Controller" is to be discontinued please take the following steps.

1. Disable/delete all the triggers assosiated with this project. (chapter 4-part 5).
2. Delete the "Virtual Mileage Logs Controller (VMLC)" file.

Chapter 7 - Source Code For this Project

Part 1 – VMLC Source Google apps script

This code can be found in the “Car Logs Google Apps Script FINAL.txt” file.

```
//Virtual Mileage Logs Controller (VMLC) source code
//Written by Elder Colby James Ransom of the Idaho Boise Mission (2019-2021)
//fourLeafBacon - (YouTube)
//handsome_ransom_productions - Instagram
//11/11/2021 project completion

// User Available Scripts
function pullAllData() {
  //Pulls a current/old Report for the specified month
  // Set the Active Spreadsheet so we don't forget
  SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
  var master = SpreadsheetApp.getActive();

  //check DEPENDANCIES
  var monthEmpty = master.getRange('A3').getValue(); // month referenced in scripts
  var yearEmpty = master.getRange('B3').getValue(); // Year referenced in scripts
  if (monthEmpty != ""){
    if (yearEmpty != ""){
      //clear the current report sheet
      master.getSheetByName("Current Report").getRange("A4:L1000").setValue("");

      //get the month we want to pull
      var monthToPull = master.getRange("A3").getValues() +'-'+ master.getRange("B3").getValues();

      //find the folder that contains the actual logs
      var masterId = master.getId();
      var masterFile = DriveApp.getFileById(masterId);
      var masterFolder = masterFile.getParents();
      var childFolder = masterFolder.next().getFolders().next();

      //get a list of mileage logs
      var file = childFolder.getFiles();

      //var to tell us how many sheets did not exist out of the files
      var notExist = 0;

      //var to keep track of iteration
      var i = 4;

      //loop through all mileage logs
      while (file.hasNext()) {//If there is a next file, then continue looping

        //get the file id to pull data
        var currentLog = file.next();
        var logId = currentLog.getId();

        //check to see if the log sheet we want exists
        var logSheets = SpreadsheetApp.openById(logId).getSheets();

        //boolian var to represent if log sheet exists
        var sheetExists = false;

        for(var w = 0; w<logSheets.length; w++){
```

```
          if(logSheets[w].getName()==monthToPull){
            var sheetExists = true;
            break;
          }
        }
        if(sheetExists == true){
          //pull the data we want
          var carNumber = SpreadsheetApp.openById(logId).getSheetByName("Car Number").getRange("A1").getValue(); //car
number

          var sheetToPullFrom = SpreadsheetApp.openById(logId).getSheetByName(monthToPull);
          var makeModel = sheetToPullFrom.getRange("C2").getValue();
          var driver = sheetToPullFrom.getRange("C4").getValue();
          var endodometer = sheetToPullFrom.getRange("A39").getValue();
          var startodometer = sheetToPullFrom.getRange("B6").getValue();
          var vin = sheetToPullFrom.getRange("H2").getValue();
          var licensePlate = sheetToPullFrom.getRange("I3").getValue();
          var startDate = sheetToPullFrom.getRange("K2").getValue();
          var area = sheetToPullFrom.getRange("K4").getValue();
          var milesDriven = sheetToPullFrom.getRange("J39").getValue();
          var mileAllottment = sheetToPullFrom.getRange("E5:E6").getValue();

          //If not the end of the month show estimated calculations
          if(!endodometer){
            var endodometer = "Driver log incomplete currently ~ " + (startodometer + sheetToPullFrom.getRange("D37").getValue());
          }
          if(!milesDriven){
            var milesDriven = (sheetToPullFrom.getRange("D37").getValue());
          }

          //push the data to our master sheet
          master.getSheetByName("Current
Report").getRange("A"+i+":L"+i).setValues([[carNumber,milesDriven,mileAllottment,startDate,area,startodometer,endodometer,driv
er,makeModel,licensePlate,vin,currentLog.getUrl()]]);

        }else{
          master.getSheetByName("Current Report").getRange("A"+i).setValue("Sheet '"+monthToPull+"' does not exist in file
'"+currentLog.getName()+"'.");
          master.getSheetByName("Current Report").getRange("F"+i).setValue(currentLog.getUrl());
          Logger.log("Sheet '"+monthToPull+"' does not exist in '"+currentLog.getName()+"'.");
          notExist++;
        }
        //increase the index by 1
        i++;

      }
      if(notExist>0){
        //var ui = SpreadsheetApp.getUi();
        Logger.log(notExist+" files did not have the sheet "+monthToPull+ ". Check 'Current Report' for details.");
      }
      Logger.log("end of pull all data");

    }else{
      var ui = SpreadsheetApp.getUi();
      ui.alert("Year referenced in scripts ('B3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
    }
  }else{
    var ui = SpreadsheetApp.getUi();
    ui.alert("Month referenced in scripts ('A3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
  }
```

```
}


function pushData() {
 //pushes the updated assosiated car/driver/information in "Master Data" to all the Driver Logs
 // Set the Active Spreadsheet so we don't forget
 SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
 var master = SpreadsheetApp.getActive();

 var monthEmpty = master.getRange('A3').getValue(); // month referenced in scripts
 var yearEmpty = master.getRange('B3').getValue(); // Year referenced in scripts
 if (monthEmpty != ""){
  if (yearEmpty != ""){
    //get the month/sheet we want to interact with
    var monthToPull = master.getRange("A3").getValues() +'-'+ master.getRange("B3").getValues();

    //get the Start date for this log
    var startDate = master.getRange("A3").getValues() +'/1/'+ master.getRange("B3").getValues();

    //find the folder that contains the actual logs
    var masterId = master.getId();
    var masterFile = DriveApp.getFileById(masterId);
    var masterFolder = masterFile.getParents();
    var childFolder = masterFolder.next().getFolders().next();

    //Get the car log index to find info
    var carLogIndex = SpreadsheetApp.openById(master.getId()).getSheetByName("Master Data").getRange("F:F").getValues();

    //get a list of mileage logs
    var file = childFolder.getFiles();

    //loop through all mileage logs
    while (file.hasNext()) {//If there is a next file, then continue looping

      //get the file id to pull data
      var currentLog = file.next();
      var logId = currentLog.getId();

      //check to see if the log sheet we want exists
      var logSheets = SpreadsheetApp.openById(logId).getSheets();

      //boolian var to represent if log sheet exists
      var sheetExists = false;

      for(var w = 0; w<logSheets.length; w++){
        if(logSheets[w].getName()==monthToPull){
          var sheetExists = true;
          break;
        }
      }
      if(sheetExists == true){

        //Find Current Car log we are working with
        var carNumber = SpreadsheetApp.openById(logId).getSheetByName("Car Number").getRange("A1").getValue();

        //Find what line the right car is on.
        for(var i=0; carLogIndex[i] != carNumber; i++){
           if (i>=carLogIndex.length){
           break;
           }
```

```
        }
      var index = (i+1);

      //get all the car info to push
      var carInfo = SpreadsheetApp.openById(master.getId()).getSheetByName("Master
Data").getRange("A"+index+":"+"P"+index).getValues();
      //Logger.log(carInfo[0][2]);

      //boolian to represent if the car was found
      var carFound = true;

      //Find what line of the index the right car is on.
      for(var i=0; carLogIndex[i] != carNumber; i++){
      if (i>=carLogIndex.length){
       var carFound = false;
       break;
       }
      }

      //if the car was not found we need not waste resources
      if (carFound == true){
       //check if there is more than one designated driver
       if(carInfo[0][13]){
        var driver = carInfo[0][12] + ", " + carInfo[0][13];
       }else{
        var driver = carInfo[0][12];
       }

       //check if there is more than one area
       if(carInfo[0][13]){
        var area = carInfo[0][2] + ", " + carInfo[0][3];
       }else{
        var area = carInfo[0][2];
       }
       //Write to the Current Car Log with the newly Updated Information
       var currentLogToPush = SpreadsheetApp.openById(logId).getSheetByName(monthToPull);
       currentLogToPush.getRange("C2").setValue(carInfo[0][6] +" "+ carInfo[0][7]); //makeModel
       currentLogToPush.getRange("C4").setValue(driver); //driver
       currentLogToPush.getRange("E5").setValue(carInfo[0][10]); //Allotted Miles
       currentLogToPush.getRange("H2").setValue(carInfo[0][4]); //vin
       currentLogToPush.getRange("I3").setValue(carInfo[0][9]); //licensePlate
       currentLogToPush.getRange("K2").setValue(startDate); //startDate
       currentLogToPush.getRange("K4").setValue(area); //area
       currentLogToPush.getRange("I4").setValue(carNumber); //carNumber
      }else{
       Logger.log("The information for car # "+carNumber+" was not found in 'Master Data' - skipping...");
      }
     }
    }
   }else{
    var ui = SpreadsheetApp.getUi();
    ui.alert("Year referenced in scripts ('B3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
   }
  }else{
   var ui = SpreadsheetApp.getUi();
   ui.alert("Month referenced in scripts ('A3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
  }


}
```

```
function pushDataForSingleLog() {
 //pushes the updated assosiated car/driver/information in "Master Data" to all the Driver Logs
 // Set the Active Spreadsheet so we don't forget
 SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
 var master = SpreadsheetApp.getActive();

 //check DEPENDANCIES
 var monthEmpty = master.getRange('A3').getValue(); // month referenced in scripts
 var yearEmpty = master.getRange('B3').getValue(); // Year referenced in scripts
 var emailEmpty = master.getRange('C3:D3').getValue(); // email referenced in scripts
 if (monthEmpty != ""){
  if (yearEmpty != ""){
   if (emailEmpty != ""){
    //declares the UI as a variable to tell errors
    var ui = SpreadsheetApp.getUi();
    //ui.alert();

    //get the month/sheet we want to interact with
    var monthToPull = master.getRange("A3").getValues() +'-'+ master.getRange("B3").getValues();

    //get the due date for this log
    var startDate = master.getRange("A3").getValues() +'/1/'+ master.getRange("B3").getValues();

    //get the car log we are wanting to push data to
    var wantedCarNumber = master.getRange("E3").getValue();

    //find the folder that contains the actual logs
    var masterId = master.getId();
    var masterFile = DriveApp.getFileById(masterId);
    var masterFolder = masterFile.getParents();
    var childFolder = masterFolder.next().getFolders().next();

    //Get the car log index to find info
    var carLogIndex = SpreadsheetApp.openById(master.getId()).getSheetByName("Master Data").getRange("F:F").getValues();

    //boolian to represent if the car was found
    var carFound = true;

    //Find what line of the index the right car is on.
    for(var e=0; carLogIndex[e] != wantedCarNumber; e++){
     if (e>=carLogIndex.length){
      var carFound = false;
      break;
     }
    }
    var index = (e+1);

    Logger.log("Car found "+carFound);
    //if the car was not found we need not waste resources
    if (carFound == true){

     //get a list of mileage logs
     var file = childFolder.getFiles();

     //loop through all mileage logs
     while (file.hasNext()) {//If there is a next file, then continue looping

      //get the file id to pull data
      var currentLog = file.next();
      var logId = currentLog.getId();
```

```
          //Logger.log(logId);
          //check to see if the log sheet we want exists
          var logSheets = SpreadsheetApp.openById(logId).getSheets();

          //boolian var to represent if log sheet exists
          var sheetExists = false;

          for(var w = 0; w<logSheets.length; w++){
           if(logSheets[w].getName()==monthToPull){
            var sheetExists = true;
            break;
           }
          }

          if(sheetExists == true){

           //Find Current Car log we are working with
           var carNumber = SpreadsheetApp.openById(logId).getSheetByName("Car Number").getRange("A1").getValue();

           if (carNumber == wantedCarNumber){
            var carInfo = SpreadsheetApp.openById(master.getId()).getSheetByName("Master
Data").getRange("A"+index+":P"+index).getValues();

            //check if there is more than one designated driver
            if(carInfo[0][13]){
             var driver = carInfo[0][12] + ", " + carInfo[0][13];
            }else{
             var driver = carInfo[0][12];
            }

            //check if there is more than one area
            if(carInfo[0][3]){
             var area = carInfo[0][2] + ", " + carInfo[0][3];
            }else{
             var area = carInfo[0][2];
            }

            Logger.log(carInfo[0]);
            //Write to the Current Car Log with the newly Updated Information
            var currentLogToPush = SpreadsheetApp.openById(logId).getSheetByName(monthToPull);
            currentLogToPush.getRange("C2").setValue(carInfo[0][6] +" "+ carInfo[0][7]); //makeModel
            currentLogToPush.getRange("C4").setValue(driver); //driver
            currentLogToPush.getRange("E5").setValue(carInfo[0][10]); //Allotted Miles
            currentLogToPush.getRange("H2").setValue(carInfo[0][4]); //vin
            currentLogToPush.getRange("I3").setValue(carInfo[0][9]); //licensePlate
            currentLogToPush.getRange("K2").setValue(startDate); //startDate
            currentLogToPush.getRange("K4").setValue(area); //area
            currentLogToPush.getRange("I4").setValue(carNumber); //carNumber

            Logger.log("ibreak after pushing data");
            break;
           }
          }else{
           Logger.log("Car # "+carNumber+" is not car # "+wantedCarNumber);
          }
         }
        }else{
         Logger.log("The information for car # "+carNumber+" was not found in 'Master Data' - skipping...");
         ui.alert("That car was not found in 'Master Data'!");
        }
```

```
    }else{
      var ui = SpreadsheetApp.getUi();
      ui.alert("Email Referenced in Scripts     ('C3:D3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
    }
   }else{
    var ui = SpreadsheetApp.getUi();
    ui.alert("Year referenced in scripts ('B3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
   }
  }else{
   var ui = SpreadsheetApp.getUi();
   ui.alert("Month referenced in scripts ('A3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
  }
}


function emailAllReportsLandscape() {
 //emails a pdf of the specified month to the specified email in Landscape format
 // Set the Active Spreadsheet so we don't forget
 SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
 var master = SpreadsheetApp.getActive();

 //check DEPENDANCIES
 var monthEmpty = master.getRange('A3').getValue(); // month referenced in scripts
 var yearEmpty = master.getRange('B3').getValue(); // Year referenced in scripts
 var emailEmpty = master.getRange('C3:D3').getValue(); // email referenced in scripts
 if (monthEmpty != ""){
  if (yearEmpty != ""){
   if (emailEmpty != ""){
    //get the month we want to pull
    var monthToPull = master.getRange("A3").getValue() +'-'+ master.getRange("B3").getValue();

    //get the email to send it to
    var emailToSendTo = master.getRange("C3").getValue();

    //find the folder that contains the actual logs
    var masterId = master.getId();
    var masterFile = DriveApp.getFileById(masterId);
    var masterFolder = masterFile.getParents();
    var childFolder = masterFolder.next().getFolders().next();

    //get a list of mileage logs
    var file = childFolder.getFiles();

    //var to tell us how many sheets did not exist out of the files
    var notExist = 0;

    //loop through all mileage logs
    while (file.hasNext()) {//If there is a next file, then continue looping

     //get the file id to pull data
     var currentLog = file.next();
     var logId = currentLog.getId();

     //check to see if the log sheet we want exists
     var logSheets = SpreadsheetApp.openById(logId).getSheets();

     //boolian var to represent if log sheet exists
     var sheetExists = false;

     for(var w = 0; w<logSheets.length; w++){
      if(logSheets[w].getName()==monthToPull){
```

```
            var sheetExists = true;
            break;
           }
          }
          if(sheetExists == true){

           //Get data fields for email
           var carNumber = SpreadsheetApp.openById(logId).getSheetByName(monthToPull).getRange("I4").getValue();
           var area = SpreadsheetApp.openById(logId).getSheetByName(monthToPull).getRange("K4").getValue();
           var makeModel = SpreadsheetApp.openById(logId).getSheetByName(monthToPull).getRange("C2").getValue();

           //hide all sheets except the one we want
           var sheets = SpreadsheetApp.openById(logId).getSheets();
           for(var i =0;i<sheets.length;i++){
            if(sheets[i].getName()!=monthToPull){
              sheets[i].hideSheet();
            }else{
              sheets[i].showSheet();
           }
           }

           // Set the text body to attach to the email.
           var body = "Mile logs for car # "+ carNumber +" "+ makeModel +" in area "+ area +".";

           // Construct the Subject Line
           var subject = "Car logs for car # "+ carNumber +" in area "+ area +" for "+ monthToPull;

           var url = "https://docs.google.com/spreadsheets/d/"+logId+"/export"+"?format=pdf&"+"portrait=false&";

           var params = {method:"GET",headers:{"authorization":"Bearer "+ ScriptApp.getOAuthToken()}};

           var response = UrlFetchApp.fetch(url, params).getBlob();

           MailApp.sendEmail(emailToSendTo, subject, body, {
              attachments: [{
                 fileName: 'Milage Logs for Car '+ carNumber + ' in area '+ area + '.pdf',
                 content: response.getBytes(),
                 mimeType: "application/pdf"
              }]
           });
          }else{
           notExist++;
           Logger.log("Sheet '"+monthToPull+ "' is not in file '"+currentLog.getName()+"'. NOT EMAILING A REPORT FOR THIS
LOG!");
          }
         }
         if(notExist>0){
          //var ui = SpreadsheetApp.getUi();
          //ui.alert(notExist+" files did not have the sheet "+monthToPull);
          Logger.log(notExist+" files did not have the sheet "+monthToPull);
         }
         Logger.log("end of 'emailAllReportsPortrait'");
        }else{
         var ui = SpreadsheetApp.getUi();
         ui.alert("Email Referenced in Scripts     ('C3:D3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
        }
       }else{
        var ui = SpreadsheetApp.getUi();
        ui.alert("Year referenced in scripts ('B3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
       }
      }else{
```

```
   var ui = SpreadsheetApp.getUi();
   ui.alert("Month referenced in scripts ('A3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
  }
}


function emailAllReportsPortrait() {
 //emails a pdf of the specified month to the specified email in Portrait format
 // Set the Active Spreadsheet so we don't forget
 SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
 var master = SpreadsheetApp.getActive();

 //check DEPENDANCIES
 var monthEmpty = master.getRange('A3').getValue(); // month referenced in scripts
 var yearEmpty = master.getRange('B3').getValue(); // Year referenced in scripts
 var emailEmpty = master.getRange('C3:D3').getValue(); // email referenced in scripts
 if (monthEmpty != ""){
  if (yearEmpty != ""){
   if (emailEmpty != ""){
    //get the month we want to pull
    var monthToPull = master.getRange("A3").getValue() +'-'+ master.getRange("B3").getValue();

    //get the email to send it to
    var emailToSendTo = master.getRange("C3").getValue();

    //find the folder that contains the actual logs
    var masterId = master.getId();
    var masterFile = DriveApp.getFileById(masterId);
    var masterFolder = masterFile.getParents();
    var childFolder = masterFolder.next().getFolders().next();

    //get a list of mileage logs
    var file = childFolder.getFiles();

    //var to tell us how many sheets did not exist out of the files
    var notExist = 0;

    //loop through all mileage logs
    while (file.hasNext()) {//If there is a next file, then continue looping

     //get the file id to pull data
     var currentLog = file.next();
     var logId = currentLog.getId();

     //check to see if the log sheet we want exists
     var logSheets = SpreadsheetApp.openById(logId).getSheets();

     //boolian var to represent if log sheet exists
     var sheetExists = false;

     for(var w = 0; w<logSheets.length; w++){
      if(logSheets[w].getName()==monthToPull){
       var sheetExists = true;
       break;
      }
     }
     if(sheetExists == true){

      //Get data fields for email
      var carNumber = SpreadsheetApp.openById(logId).getSheetByName(monthToPull).getRange("I4").getValue();
      var area = SpreadsheetApp.openById(logId).getSheetByName(monthToPull).getRange("K4").getValue();
```

```
          var makeModel = SpreadsheetApp.openById(logId).getSheetByName(monthToPull).getRange("C2").getValue();

          //hide all sheets except the one we want
          var sheets = SpreadsheetApp.openById(logId).getSheets();
          for(var i =0;i<sheets.length;i++){
            if(sheets[i].getName()!=monthToPull){
              sheets[i].hideSheet();
            }else{
              sheets[i].showSheet();
            }
          }

          // Set the text body to attach to the email.
          var body = "Mile logs for car # "+ carNumber +" "+ makeModel +" in area "+ area +".";

          // Construct the Subject Line
          var subject = "Car logs for car # "+ carNumber +" in area "+ area +" for "+ monthToPull;

          // Make the pdf
          var pdf = DriveApp.getFileById(logId).getAs('application/pdf').getBytes();
          var attach = {fileName:'Milage Logs for Car '+ carNumber + ' in area '+ area + '.pdf',content:pdf, mimeType:'application/pdf'};

          // Send the freshly constructed email
          MailApp.sendEmail(emailToSendTo, subject, body, {attachments:[attach]});
        }else{
          notExist++;
          Logger.log("Sheet '"+monthToPull+ "' is not in file '"+currentLog.getName()+"'. NOT EMAILING A REPORT FOR THIS
LOG!");
        }
      }
      if(notExist>0){
        //var ui = SpreadsheetApp.getUi();
        //ui.alert(notExist+" files did not have the sheet "+monthToPull);
        Logger.log(notExist+" files did not have the sheet "+monthToPull);
      }
      Logger.log("end of 'emailAllReportsPortrait'");
    }else{
      var ui = SpreadsheetApp.getUi();
      ui.alert("Email Referenced in Scripts     ('C3:D3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
    }
  }else{
    var ui = SpreadsheetApp.getUi();
    ui.alert("Year referenced in scripts ('B3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
  }
 }else{
  var ui = SpreadsheetApp.getUi();
  ui.alert("Month referenced in scripts ('A3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
 }

}


function createNewCarLog(){
 //Creates a new car log from the car number put in the specified cell, it also includes the updated information for said car in the
"Master Data" sheet.
 // Set the Active Spreadsheet so we don't forget
 SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
 var master = SpreadsheetApp.getActive();

 //CHECK DEPENDANCIES
 var monthEmpty = master.getRange('A3').getValue(); // month referenced in scripts
```

```
 var yearEmpty = master.getRange('B3').getValue(); // Year referenced in scripts
 var carNumberEmpty = master.getRange('E3:F3').getValue(); // Car Number referenced in scripts
 if (monthEmpty != ""){
  if (yearEmpty != ""){
   if (carNumberEmpty != ""){
    //prepare the log template to overwrite new log
    var newLogTemplate = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Log Template");

    //get the car log number so we know what new car log we are creating
    var newCarNumber = master.getRange("E3").getValue();

    //Get the car log index to find the proper car info
    var carLogIndex = SpreadsheetApp.openById(master.getId()).getSheetByName("Master Data").getRange("F:F").getValues();

    //if the car was not found message
    var carNotFound = "";

    //Find what line of the index the right car is on.
    for(var i=0; carLogIndex[i] != newCarNumber; i++){
     if (i>=carLogIndex.length){
      var carNotFound = "CAR WAS NOT IN DATABASE!";
      break;
     }
    }
    var index = (i+1);

    //get all the car info to push
    var carInfo = SpreadsheetApp.openById(master.getId()).getSheetByName("Master
Data").getRange("A"+index+":"+"P"+index).getValues();
    //Logger.log(carInfo[0][2]);

    //get our designated Drivers
    if(carInfo[0][13]){
     var designatedDriver = carInfo[0][12] + ", " + carInfo[0][13];
    }else{
     var designatedDriver = carInfo[0][12];
    }

    //get the area/areas assigned to car
    if(carInfo[0][3]){
     var area = carInfo[0][2] + ", " + carInfo[0][3];
    }else{
     var area = carInfo[0][2];
    }


    //get the month/sheet we want to create
    var monthToPull = master.getRange("A3").getValues() +'-'+ master.getRange("B3").getValues();
    var logStartDate = master.getRange("A3").getValues() +'/1/'+ master.getRange("B3").getValues();

    //find the folder that contains the other logs
    var masterId = master.getId();
    var masterFile = DriveApp.getFileById(masterId);
    var masterFolder = masterFile.getParents();
    var childFolder = masterFolder.next().getFolders().next();

    //create log and move it to where the other logs are.
    var newLog = SpreadsheetApp.create("Car # " + newCarNumber + " " + carInfo[0][6] + " "+ carInfo[0][7]+carNotFound, 42,
12);
    var newLogId = newLog.getId();
    var newLogFile = DriveApp.getFileById(newLogId);
```

```
      newLogFile.moveTo(childFolder);

      //Copy log template to the new log that was created
      var newCarLog = SpreadsheetApp.openById(newLogId);
      newLogTemplate.copyTo(newCarLog);

      //Remove the default extra sheet that will not be used
      var defaultSheet = newCarLog.getSheetByName("Sheet1");
      newCarLog.deleteSheet(defaultSheet);

      //Rename the template sheet to the proper name
        newLog.getSheets()[0].setName(monthToPull);

      //creates another sheet that only contains the car number
      var carNumberSheet = SpreadsheetApp.openById(newLogId).insertSheet("Car Number");

      //delete all cells except a1 on the car number sheet
      var lastCol = carNumberSheet.getMaxColumns()-1;
      var lastRow = carNumberSheet.getMaxRows()-2;
      carNumberSheet.deleteColumns(2, lastCol);
      carNumberSheet.deleteRows(2, lastRow);

      //push car number to the car number sheet
      carNumberSheet.getRange("A1").setValue(carInfo[0][5]); //car number
      carNumberSheet.getRange("A2").setValue("DO NOT DELETE THIS SHEET"); //car number
      carNumberSheet.hideSheet();

      //Add proper information to the new Log
      if(carInfo[0][0]){
      var sheetToPushTo = SpreadsheetApp.openById(newLogId).getSheetByName(monthToPull);
      sheetToPushTo.getRange("C2").setValue(carInfo[0][6] + " "+ carInfo[0][7]); //make model
      sheetToPushTo.getRange("H2").setValue(carInfo[0][4]); //vin
      sheetToPushTo.getRange("K2").setValue(logStartDate); //Log Start Date
      sheetToPushTo.getRange("I3").setValue(carInfo[0][9]); //plate Number
      sheetToPushTo.getRange("C4").setValue(designatedDriver); //Designated Driver
      sheetToPushTo.getRange("I4").setValue(newCarNumber); //car number
      sheetToPushTo.getRange("K4").setValue(area); //area car is assigned
      sheetToPushTo.getRange("E5").setValue(carInfo[0][10]); //allotted miles
       }
     }else{
       var ui = SpreadsheetApp.getUi();
       ui.alert("Car Number Referenced in scripts ('E3:F3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
     }
    }else{
      var ui = SpreadsheetApp.getUi();
      ui.alert("Year referenced in scripts ('B3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
    }
  }else{
    var ui = SpreadsheetApp.getUi();
    ui.alert("Month referenced in scripts ('A3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
  }

}


function createAllNewCarLogs(){
  //Creates a log for every car in the "Master Data" sheet.
  // Set the Active Spreadsheet so we don't forget
  var MasterSheet = SpreadsheetApp.getActiveSpreadsheet();

  //CHECK DEPENDANCIES
```

```
var monthEmpty = MasterSheet.getRange('A3').getValue(); // month referenced in scripts
var yearEmpty = MasterSheet.getRange('B3').getValue(); // Year referenced in scripts
if (monthEmpty != ""){
  if (yearEmpty != ""){
    //set the script sheet in the active spreadsheet
    var scriptMasterSheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet");

    //set the master data sheet in the active spreadsheet
    var masterData = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Master Data");

    //prepare the log template to overwrite new log
    var newLogTemplate = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Log Template");

    //Get the car log index to find the proper car info
    var carLogIndex = masterData.getRange("F:F").getValues();

    //get the month/sheet we want to create
    var monthToPull = scriptMasterSheet.getRange("A3").getValues() +'-'+ scriptMasterSheet.getRange("B3").getValues();
    var logStartDate = scriptMasterSheet.getRange("A3").getValues() +'/1/'+ scriptMasterSheet.getRange("B3").getValues();

    //find the folder that contains the other logs
    var masterId = MasterSheet.getId();
    var masterFile = DriveApp.getFileById(masterId);
    var masterFolder = masterFile.getParents();
    var childFolder = masterFolder.next().getFolders().next();

    //Loop through all the cars listed in the "Master Data" sheet
    for(var i=2; carLogIndex[i]!=""; i++){
      var index = (i+1);
      //get all the car info to push
      var carInfo = masterData.getRange("A"+index+":P"+index).getValues();
      Logger.log(carInfo);
      //get our designated Drivers
      if(carInfo[0][13]){
        var designatedDriver = carInfo[0][12] + ", " + carInfo[0][13];
      }else{
        var designatedDriver = carInfo[0][12];
      }

      //get the area/areas assigned to car
      if(carInfo[0][3]){
        var area = carInfo[0][2] + ", " + carInfo[0][3];
      }else{
        var area = carInfo[0][2];
      }

      //create log and move it to where the other logs are.
      var newLog = SpreadsheetApp.create("Car # " + carInfo[0][5] + " " + carInfo[0][6] + " "+ carInfo[0][7], 42, 12);
      var newLogId = newLog.getId();
      var newLogFile = DriveApp.getFileById(newLogId);
      newLogFile.moveTo(childFolder);

      //Copy log template to the new log that was created
      var newCarLog = SpreadsheetApp.openById(newLogId);
      newLogTemplate.copyTo(newCarLog);

      //Remove the default extra sheet that will not be used
      var defaultSheet = newCarLog.getSheetByName("Sheet1");
      newCarLog.deleteSheet(defaultSheet);

      //Rename the template sheet to the proper name
```

```
      newLog.getSheets()[0].setName(monthToPull);

    //creates another sheet that only contains the car number
    var carNumberSheet = SpreadsheetApp.openById(newLogId).insertSheet("Car Number");

    //delete all cells except a1 on the car number sheet
    var lastCol = carNumberSheet.getMaxColumns()-1;
    var lastRow = carNumberSheet.getMaxRows()-2;
    carNumberSheet.deleteColumns(2, lastCol);
    carNumberSheet.deleteRows(2, lastRow);

    //push car number to the car number sheet
    carNumberSheet.getRange("A1").setValue(carInfo[0][5]); //car number
    carNumberSheet.getRange("A2").setValue("DO NOT DELETE THIS SHEET"); //car number
    carNumberSheet.hideSheet();

      //Add proper information to the new Log
      if(carInfo[0][0]){
      newCarLog.getSheetByName(monthToPull).getRange("C2").setValue(carInfo[0][6] + " "+ carInfo[0][7]); //make model
      newCarLog.getSheetByName(monthToPull).getRange("H2").setValue(carInfo[0][4]); //vin
      newCarLog.getSheetByName(monthToPull).getRange("K2").setValue(logStartDate); //Log Start Date
      newCarLog.getSheetByName(monthToPull).getRange("I3").setValue(carInfo[0][9]); //plate Number
      newCarLog.getSheetByName(monthToPull).getRange("C4").setValue(designatedDriver); //Designated Driver
      newCarLog.getSheetByName(monthToPull).getRange("I4").setValue(carInfo[0][5]); //car number
      newCarLog.getSheetByName(monthToPull).getRange("K4").setValue(area); //area car is assigned
      newCarLog.getSheetByName(monthToPull).getRange("E5").setValue(carInfo[0][10]); //allotted miles
      }
     }
    }else{
     var ui = SpreadsheetApp.getUi();
     ui.alert("Year referenced in scripts ('B3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
    }
   }else{
    var ui = SpreadsheetApp.getUi();
    ui.alert("Month referenced in scripts ('A3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
   }
}


function shareWithProperAreaEmails() {
  //reshares/shares all the logs with their specified area email
  // Set the Active Spreadsheet so we don't forget
  SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
  var master = SpreadsheetApp.getActive();

  //CHECK DEPENDANCIES
  var monthEmpty = master.getRange('A3').getValue(); // month referenced in scripts
  var yearEmpty = master.getRange('B3').getValue(); // Year referenced in scripts
  if (monthEmpty != ""){
   if (yearEmpty != ""){
     //find the folder that contains the actual logs
     var masterId = master.getId();
     var masterFile = DriveApp.getFileById(masterId);
     var masterFolder = masterFile.getParents();
     var childFolder = masterFolder.next().getFolders().next();

     //get a list of mileage logs
     var file = childFolder.getFiles();

     //Find the row for said car log
     var carLogIndex = SpreadsheetApp.openById(master.getId()).getSheetByName("Master Data").getRange("F:F").getValues();
```

```
    //loop through all mileage logs
    while (file.hasNext()) {//If there is a next file, then continue looping

      //get the file id to pull data
      var currentLog = file.next();
      var logId = currentLog.getId();

      //Find Current Car log we are working with
      var carNumber = SpreadsheetApp.openById(logId).getSheetByName("Car Number").getRange("A1").getValue();

      //tell me what car we are on
      Logger.log("Car # " + carNumber);

      //Find what line of the index the right car is on.
      for(var i=0; carLogIndex[i] != carNumber; i++){
         if (i>=carLogIndex.length){
           Logger.log("Did not find the car in the list!");
           break;
         }
       }
      var index = (i+1);

      //get the area emails for the specified mile log
      var areaEmails = SpreadsheetApp.openById(master.getId()).getSheetByName("Master
Data").getRange("O"+index+":P"+index).getValues();

      //get editors and owner of current log
      var editors = currentLog.getEditors();
      var owner = currentLog.getOwner();

      //Remove all editors
      for(var i=0; i<editors.length; i++){
        if(editors[i]!=owner[0]){
          currentLog.removeEditor(editors[i]);
        }
      }

      //add the appropriate Area emails
      currentLog.addEditor(areaEmails[0][0]).setShareableByEditors(false);
      if(areaEmails[0][1]){
        currentLog.addEditor(areaEmails[0][1]).setShareableByEditors(false);
      }

    }
    protectSheets();
   }else{
    var ui = SpreadsheetApp.getUi();
    ui.alert("Year referenced in scripts ('B3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
   }
  }else{
   var ui = SpreadsheetApp.getUi();
   ui.alert("Month referenced in scripts ('A3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
  }
}


function shareSingleLogWithProperAreaEmails() {
  //reshares/shares a specified log with its associated area email.
  // Set the Active Spreadsheet so we don't forget
  SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
```

```
var master = SpreadsheetApp.getActive();

var monthEmpty = master.getRange('A3').getValue(); // month referenced in scripts
var yearEmpty = master.getRange('B3').getValue(); // Year referenced in scripts
var carNumberEmpty = master.getRange('E3:F3').getValue(); // Car Number referenced in scripts
if (monthEmpty != ""){
  if (yearEmpty != ""){
    if (carNumberEmpty != ""){
      //get the car we are sharing
      var carToShare = master.getRange("E3").getValue();

      //make sure we don't waste our time
      if(carToShare!=""){

        //find the folder that contains the actual logs
        var masterId = master.getId();
        var masterFile = DriveApp.getFileById(masterId);
        var masterFolder = masterFile.getParents();
        var childFolder = masterFolder.next().getFolders().next();

        //get a list of mileage logs
        var file = childFolder.getFiles();

        //Find the row for said car log
        var carLogIndex = SpreadsheetApp.openById(master.getId()).getSheetByName("Master Data").getRange("F:F").getValues();

        //loop through all mileage logs
        while (file.hasNext()) {//If there is a next file, then continue looping

          //get the file id to pull data
          var currentLog = file.next();
          var logId = currentLog.getId();

          //Find Current Car log we are working with
          var carNumber = SpreadsheetApp.openById(logId).getSheetByName("Car Number").getRange("A1").getValue();

          //tell me what car we are currently looking at
          Logger.log("Car # " + carNumber);

          if(carNumber == carToShare){
            //Find what line of the index the right car is on.
          for(var i=0; carLogIndex[i] != carNumber; i++){
            if (i>=carLogIndex.length){
              Logger.log("The car was not found in the index!")
              break;
              }
            }
            var index = (i+1);

            //get the area emails for the specified mile log
            var areaEmails = SpreadsheetApp.openById(master.getId()).getSheetByName("Master
Data").getRange("O"+index+":P"+index).getValues();

            //get editors and owner of current log
            var editors = currentLog.getEditors();
            var owner = currentLog.getOwner();

            //Remove all editors
            for(var i=0; i<editors.length; i++){
              if(editors[i]!=owner[0]){
                currentLog.removeEditor(editors[i]);
```

```
            }
           }

           //add the appropriate Area emails
           currentLog.addEditor(areaEmails[0][0]).setShareableByEditors(false);
           if(areaEmails[0][1]){
            currentLog.addEditor(areaEmails[0][1]).setShareableByEditors(false);
           }

           //after the specified car is found, stop keep looking for it.
           break;
          }
         }
         protectSingleSpreadsheet();
        }
       }else{
        var ui = SpreadsheetApp.getUi();
        ui.alert("Car Number Referenced in scripts ('E3:F3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
       }
     }else{
      var ui = SpreadsheetApp.getUi();
      ui.alert("Year referenced in scripts ('B3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
     }
   }else{
    var ui = SpreadsheetApp.getUi();
    ui.alert("Month referenced in scripts ('A3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
   }
}


function unshareAllRepoprts() {
  //Removes all area email access to all logs. For repairs or emergency only.
  // Set the Active Spreadsheet so we don't forget
  SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
  var master = SpreadsheetApp.getActive();

  //find the folder that contains the actual logs
  var masterId = master.getId();
  var masterFile = DriveApp.getFileById(masterId);
  var masterFolder = masterFile.getParents();
  var childFolder = masterFolder.next().getFolders().next();

  //get a list of mileage logs
  var file = childFolder.getFiles();

  //loop through all mileage logs
  while (file.hasNext()) {//If there is a next file, then continue looping

    //get the file id to pull data
    var currentLog = file.next();

    //get editors and owner of current log
    var editors = currentLog.getEditors();

    //Remove all editors
    for(var i=0; i<editors.length; i++){
     currentLog.removeEditor(editors[i]);
    }
  }
}
```

```
function replaceCellWith() {
 //similar to a find and replace but for all the carlogs/sheets
 // Set the Active Spreadsheet so we don't forget
 SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
 var master = SpreadsheetApp.getActive();

 //CHECK DEPENDANCIES
 var monthEmpty = master.getRange('A3').getValue(); // month referenced in scripts
 var yearEmpty = master.getRange('B3').getValue(); // Year referenced in scripts
 var replaceEmpty = master.getRange('F22').getValue(); // find and replace "Text to put in cell"
 var findEmpty = master.getRange('G22').getValue(); // find and replace "Cell to Find"
 if (monthEmpty != ""){
  if (yearEmpty != ""){
   if (replaceEmpty != ""){
    if (findEmpty != ""){
     //get the month/sheet we want to interact with
     var monthToPull = master.getRange("A3").getValues() +'-'+ master.getRange("B3").getValues();

     //get the cell to replace
     var newCellLocation = master.getRange("F22").getValue();

     //get the data that we will be pushing to the cell
     var newCellData = master.getRange("G22").getValue();

     //find the folder that contains the actual logs
     var masterId = master.getId();
     var masterFile = DriveApp.getFileById(masterId);
     var masterFolder = masterFile.getParents();
     var childFolder = masterFolder.next().getFolders().next();

     //get a list of mileage logs
     var file = childFolder.getFiles();

     //loop through all mileage logs
     while (file.hasNext()) {//If there is a next file, then continue looping

      //get the file id to pull data
      var currentLog = file.next();
      var logId = currentLog.getId();

      //check to see if the log sheet we want exists
      var logSheets = SpreadsheetApp.openById(logId).getSheets();

      //boolian var to represent if log sheet exists
      var sheetExists = false;

      for(var w = 0; w<logSheets.length; w++){
       if(logSheets[w]==monthToPull){
        var sheetExists = true;
        break;
       }
      }
      if(sheetExists == true){
      //push the data
      SpreadsheetApp.openById(logId).getSheetByName(monthToPull).getRange(newCellLocation).setValue(newCellData);
      }
     }
    }else{
     var ui = SpreadsheetApp.getUi();
     ui.alert("Cell to Find ('G22') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
```

```
      }
     }else{
      var ui = SpreadsheetApp.getUi();
      ui.alert("Text to put in cell ('F22') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
     }
    }else{
     var ui = SpreadsheetApp.getUi();
     ui.alert("Year referenced in scripts ('B3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
    }
  }else{
   var ui = SpreadsheetApp.getUi();
   ui.alert("Month referenced in scripts ('A3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
  }
}


function testForErrors (){
  //tests logs for common errors that may cause the other scripts to fail
  //gets the master spreadsheet object
  var masterSpreadsheet = SpreadsheetApp.getActiveSpreadsheet();

  //CHECK DEPENDANCIES
  var monthEmpty = master.getRange('A3').getValue(); // month referenced in scripts
  var yearEmpty = master.getRange('B3').getValue(); // Year referenced in scripts
  if (monthEmpty != ""){
   if (yearEmpty != ""){
     //declares the UI as a variable to tell errors
     var ui = SpreadsheetApp.getUi();
     //ui.alert();

     //gets the Script Master Sheet object as a variable
     try {
      var scriptMasterSheet = masterSpreadsheet.getSheetByName("Script Master Sheet");
      scriptMasterSheet.getName();
     }
     catch(err) {
      ui.alert("The 'Script Master Sheet' was not found - this is a problem... check file history to restore/recreate it.");
      var scriptMasterSheet = "";
     }

     //gets the "Master Data" sheet object as a variable
     try {
      var masterDataSheet = masterSpreadsheet.getSheetByName("Master Data");
      masterDataSheet.getName();
     }
     catch(err) {
      ui.alert("The 'Master Data' sheet was not found - this is a problem... check file history to restore/recreate it.");
      var masterDataSheet = "";
     }

     //gets the "Current Report" sheet object as a variable
     try {
      var currentReportSheet = masterSpreadsheet.getSheetByName("Current Report");
      currentReportSheet.getName();
     }
     catch(err) {
      ui.alert("The 'Current Report' sheet was not found - this is a problem... no biggie just restore it via file history");
      var currentReportSheet = "";
     }

     //gets the "Log Template" sheet object as a variable
```

```
    try {
      var logTemplateSheet = masterSpreadsheet.getSheetByName("Log Template");
      logTemplateSheet.getName();
    }
    catch(err) {
      ui.alert("The 'Log Template' sheet was not found - this is a problem... this sheet is used to create new car logs or create the next month's car log. if this sheet is corrupted or missing you are in trouble. Restore this template via file history.");
      var logTemplateSheet = "";
    }

    //gets the "List of all area emails" sheet object as a variable
    try {
      var listofallareaemailsSheet = masterSpreadsheet.getSheetByName("List of all area emails");
      listofallareaemailsSheet.getName();
    }
    catch(err) {
      ui.alert("The 'List of all area emails' sheet was not found - this is a problem if 'Master Data' does not contain the area emails you are sharing with directly inputted if this is the case just make a sheet called 'List of all area emails' then you get no more error :) if this is not the case restore it via file history.");
      var listofallareaemailsSheet = "";
    }

    //will not continue checking for other problems until the current spreadsheet is fixed
    if(scriptMasterSheet =="" || masterDataSheet =="" || currentReportSheet =="" || logTemplateSheet =="" || listofallareaemailsSheet ==""){
      ui.alert("The script will not continue checking for other problems until the current spreadsheet is fixed!");
    }else{

    //let's check and make sure all the files have a "Car Number" sheet
      //find the folder that contains the actual logs
      var masterId = masterSpreadsheet.getId();
      var masterFile = DriveApp.getFileById(masterId);
      var masterFolder = masterFile.getParents();
      var childFolder = masterFolder.next().getFolders().next();

      //get a list of mileage logs
      var file = childFolder.getFiles();

      //loop through all mileage logs
      while (file.hasNext()) {//If there is a next file, then continue looping

        //get the file id to pull data
        var currentLog = file.next();
        var logId = currentLog.getId();

        //let's check and make sure all the files have a "Car Number" sheet
        try {
          var carNumberSheet = SpreadsheetApp.openById(logId).getSheetByName("Car Number");
          Logger.log("Test if 'Car Number' exists in " + currentLog.getName());
          carNumberSheet.getName();
        }
        catch(err) {
          ui.alert("The 'Car Number' sheet was not found in log '"+currentLog.getName()+"', it contains two cells, A1 contains the car number, A2 contains a warning telling editors not to delete the sheet, There are no other cells besides those two. This sheet is referenced across almost all scripts, so if you are getting an error this is probably why. To fix this problem open log '" +currentLog.getName()+"' and add a sheet named 'Car Number'. Make the value of A1 the car number(for example '5'). Make the value of cell A2 'DO NOT DELETE THIS SHEET'. Delete all other columns and rows except those containing A1 or A2. If this is an older log remove it out of the working directory for the car logs. Create a new folder for these, just move them elsewhere idk.");
          var currentReportSheet = "";
        }
      }
```

```
    }
   }else{
    var ui = SpreadsheetApp.getUi();
    ui.alert("Year referenced in scripts ('B3') is empty!\nThis may be the problem!\nMake sure to hit enter or exit the cell before you start a script!");
   }
  }else{
   var ui = SpreadsheetApp.getUi();
   ui.alert("Month referenced in scripts ('A3') is empty!\nThis may be the problem!\nMake sure to hit enter or exit the cell before you start a script!");
  }


}


function switchCars () {
 var lock = LockService.getScriptLock();
 lock.tryLock(1000);
 //Switches cars and their assosiated car information then pushes that data shares it and protects it
 // Set the Active Spreadsheet so we don't forget
 SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
 var master = SpreadsheetApp.getActive();

 //CHECK DEPENDANCIES
 var monthEmpty = master.getRange('A3').getValue(); // month referenced in scripts
 var yearEmpty = master.getRange('B3').getValue(); // Year referenced in scripts
 if (monthEmpty != ""){
  if (yearEmpty != ""){
   //get the cars we will be switching
   var carOne = master.getRange("F33").getValue();
   var carTwo = master.getRange("G33").getValue();

   //declare master data
   var masterData = SpreadsheetApp.openById(master.getId()).getSheetByName("Master Data");

   //Get the car log index to find info
   var carLogIndex = master.getSheetByName("Master Data").getRange("F:F").getValues();

   //boolian to represent if the car was found
   var carOneFound = true;

   //Find what line of the index of car 1 is on
   for(var i=0; carLogIndex[i] != carOne; i++){
    if (i>=carLogIndex.length){
     var carOneFound = false;
     break;
    }
   }
   var carOneLine = (i+1);

   //get the data for car 1
   var carOneData = masterData.getRange("A"+carOneLine+":P"+carOneLine).getValues();

   //Find what line of the index of car 2 is on
   for(var i=0; carLogIndex[i] != carTwo; i++){
    if (i>=carLogIndex.length){
     var carOneFound = false;
     break;
    }
   }
```

```
    var carTwoLine = (i+1);

    //get the data for car 2
    var carTwoData = master.getSheetByName("Master Data").getRange("A"+carTwoLine+":P"+carTwoLine).getValues();

    Logger.log(carOneData);
    Logger.log(carTwoData);

    //move the data from one to the other - 'car 1 old' to 'car 1 new' (car 2 old)
    masterData.getRange("A"+carTwoLine).setValue(carOneData[0][0]);//zone
    masterData.getRange("B"+carTwoLine).setValue(carOneData[0][1]);//district
    masterData.getRange("C"+carTwoLine).setValue(carOneData[0][2]);//area
    masterData.getRange("D"+carTwoLine).setValue(carOneData[0][3]);//area 2
    masterData.getRange("K"+carTwoLine).setValue(carOneData[0][10]); //mile allotment
    masterData.getRange("M"+carTwoLine).setValue(carOneData[0][12]);//desinated driver 1
    masterData.getRange("N"+carTwoLine).setValue(carOneData[0][13]); //designated driver 2

    //move the data from one to the other - 'car 2 old' to 'car 2 new' (car 1 old)
    masterData.getRange("A"+carOneLine).setValue(carTwoData[0][0]);//zone
    masterData.getRange("B"+carOneLine).setValue(carTwoData[0][1]);//district
    masterData.getRange("C"+carOneLine).setValue(carTwoData[0][2]);//area
    masterData.getRange("D"+carOneLine).setValue(carTwoData[0][3]);//area 2
    masterData.getRange("K"+carOneLine).setValue(carTwoData[0][10]); //mile allotment
    masterData.getRange("M"+carOneLine).setValue(carTwoData[0][12]);//desinated driver 1
    masterData.getRange("N"+carOneLine).setValue(carTwoData[0][13]); //designated driver 2


    //push the data, share the car log, and protect the car log
    //for Car one
    master.getSheetByName("Script Master Sheet").getRange("E3").setValue(carOne);
    pushDataForSingleLog();
    shareSingleLogWithProperAreaEmails();

    //for Car Two
    master.getSheetByName("Script Master Sheet").getRange("E3").setValue(carTwo);
    pushDataForSingleLog();
    shareSingleLogWithProperAreaEmails();

    lock.releaseLock();
   }else{
    var ui = SpreadsheetApp.getUi();
    ui.alert("Year referenced in scripts ('B3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
   }
  }else{
   var ui = SpreadsheetApp.getUi();
   ui.alert("Month referenced in scripts ('A3') is empty!\nMake sure to hit enter or exit the cell before you start a script!");
  }


}




// Service Functions
function protectSheets() {
  //protects all the sheets from editing except those that are incomplete. Run after running either shareWithProperAreaEmails or
shareSingleLogWithProperAreaEmails
  // Set the Active Spreadsheet so we don't forget
  SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
  var master = SpreadsheetApp.getActive();
```

```
//get the month we want to pull
var monthToPull = master.getRange("A3").getValues() +'-'+ master.getRange("B3").getValues();

//find the folder that contains the actual logs
var masterId = master.getId();
var masterFile = DriveApp.getFileById(masterId);
var masterFolder = masterFile.getParents();
var childFolder = masterFolder.next().getFolders().next();

//get a list of mileage logs
var file = childFolder.getFiles();

//loop through all mileage logs
while (file.hasNext()) {//If there is a next file, then continue looping

 //get the file id to pull data
 var currentLog = file.next();
 var logId = currentLog.getId();

 Logger.log("");
 Logger.log(currentLog.getName());

 //get editors and owner of current log
 var editors = currentLog.getEditors();
 var owner = currentLog.getOwner().getEmail();

 //get the sheets for the current log
 var sheets = SpreadsheetApp.openById(logId).getSheets();

 //get current log in spreadsheet app
 var spreadsheetCurrent = SpreadsheetApp.openById(logId);

 //remove all protections to not create confusion
 var protections = spreadsheetCurrent.getProtections(SpreadsheetApp.ProtectionType.SHEET);
 try{
  for (var w = 0; w < protections.length; w++) {
   var protection = protections[w];
   if (protection.canEdit()) {
    protection.remove();
   }
  }
 }
 catch(err){
  //Logger.log("no full sheet protections");
 }
 try{
  var protections = spreadsheetCurrent.getProtections(SpreadsheetApp.ProtectionType.RANGE);

  for (var t = 0; t < protections.length; t++) {
   var protection = protections[t];
   if (protection.canEdit()) {
    protection.remove();
   }
  }
 }
 catch(err){
  //Logger.log("no range protections");
 }

 //unhide all sheets
```

```
for(var i =0;i<sheets.length;i++){
  sheets[i].showSheet();
}

//loop through all the sheets to apply protections and hide unused ones
for(var i =0;i<sheets.length;i++){

  //make sure we are not checking "Car Number"
  if(sheets[i].getName() == "Car Number"){

    Logger.log("The current sheet is 'Car Number'");
    //protect Car Number sheet
    var protectedSheetCarNumberSheet = sheets[i].protect();

    //disallow all editors to edit Car Number sheet
    for(var w=0; w<editors.length; w++){
      if(editors[w]!=owner){
        protectedSheetCarNumberSheet.removeEditor(editors[w]);
      }
    }

    var endodometer = "filler";

  }else{
    if(sheets[i].getName() == monthToPull){

      Logger.log("The current log is '"+monthToPull+"'");

      //protect the mile allotment cell in the sheet
      var protectedSheetmonthtopull = sheets[i].getRange("E5:E6").protect();
      var protectedScriptThings = sheets[i].getRange("C7:E37").protect();

      //disallow all editors to edit in that cell
      for(var w=0; w<editors.length; w++){
        if(editors[w]!=owner){
          protectedSheetmonthtopull.removeEditor(editors[w]);
          protectedScriptThings.removeEditor(editors[w]);
        }
      }
    }else{

      Logger.log("The current sheet is a previous month");
      //get the odometer for the current sheet
      var endodometer = SpreadsheetApp.openById(logId).getSheetByName(sheets[i].getName()).getRange("A39").getValue();

      //If the sheet is not completed allow editing
      if(endodometer == ""){
        Logger.log("The previous months sheet '"+sheets[i].getName()+"' is not complete - only protecting the mileage allotment");
        //protect the mile allotment cell in the sheet
        var protectedSheetOtherMonth = sheets[i].getRange("E5:E6").protect();

        //disallow all editors to edit in that cell
        for(var w=0; w<editors.length; w++){
          if(editors[w]!=owner){
            protectedSheetOtherMonth.removeEditor(editors[w]);
          }
        }
      }else{
        Logger.log("The previous months sheet '"+sheets[i].getName()+"' is complete - protecting entire sheet");
        //the sheet for a different month is complete - disallow editing
        //protect completed previous months sheet
```

```
          var protectedSheetCarNumberSheet = sheets[i].protect();

          //disallow all editors to edit completed previous months sheet
          for(var w=0; w<editors.length; w++){
           if(editors[w]!=owner){
            protectedSheetCarNumberSheet.removeEditor(editors[w]);
           }
          }
         }
        }


      }
      //hide all sheets except the one we want
      if(sheets[i].getName()!=monthToPull && endodometer != ""){
       sheets[i].hideSheet();
      }else{
       sheets[i].showSheet();
      }
     }

   }


}


function protectSingleSpreadsheet() {
  //protects all the sheets from editing except those that are incomplete. Run after running either shareWithProperAreaEmails or
shareSingleLogWithProperAreaEmails
  // Set the Active Spreadsheet so we don't forget
  SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
  var master = SpreadsheetApp.getActive();

  //get the month we want to pull
  var monthToPull = master.getRange("A3").getValues() +'-'+ master.getRange("B3").getValues();

  //get the car log we want to protect
  var carLogToProtect = master.getRange("E3").getValue();
  //var currentCarLogNumber = SpreadsheetApp.openById(logId).getSheetByName("Car Number").getRange("A1").getValue(); //car
number

  //find the folder that contains the actual logs
  var masterId = master.getId();
  var masterFile = DriveApp.getFileById(masterId);
  var masterFolder = masterFile.getParents();
  var childFolder = masterFolder.next().getFolders().next();

  //get a list of mileage logs
  var file = childFolder.getFiles();

  //loop through all mileage logs
  while (file.hasNext()) {//If there is a next file, then continue looping

    //get the file id to pull data
    var currentLog = file.next();
    var logId = currentLog.getId();

    //get the current car log number
    var currentCarLogNumber = SpreadsheetApp.openById(logId).getSheetByName("Car Number").getRange("A1").getValue(); //car
number
```

```
//if car log is not the one we are looking for we skip it
if(carLogToProtect == currentCarLogNumber){

  Logger.log("");
  Logger.log(currentLog.getName());

  //get editors and owner of current log
  var editors = currentLog.getEditors();
  var owner = currentLog.getOwner().getEmail();

  //get the sheets for the current log
  var sheets = SpreadsheetApp.openById(logId).getSheets();

  //get current log in spreadsheet app
  var spreadsheetCurrent = SpreadsheetApp.openById(logId);

  //remove all protections to not create confusion
  var protections = spreadsheetCurrent.getProtections(SpreadsheetApp.ProtectionType.SHEET);
  try{
    for (var w = 0; w < protections.length; w++) {
      var protection = protections[w];
      if (protection.canEdit()) {
        protection.remove();
      }
    }
  }
  catch(err){
    //Logger.log("no full sheet protections");
  }
  try{
    var protections = spreadsheetCurrent.getProtections(SpreadsheetApp.ProtectionType.RANGE);

    for (var t = 0; t < protections.length; t++) {
      var protection = protections[t];
      if (protection.canEdit()) {
        protection.remove();
      }
    }
  }
  catch(err){
    //Logger.log("no range protections");
  }

  //unhide all sheets
  for(var i =0;i<sheets.length;i++){
  sheets[i].showSheet();
  }

  //loop through all the sheets to apply protections and hide unused ones
  for(var i =0;i<sheets.length;i++){

    //unhide all sheets
    sheets[i].showSheet();

    //make sure we are not checking "Car Number"
    if(sheets[i].getName() == "Car Number"){

      Logger.log("The current sheet is 'Car Number'");
      //protect Car Number sheet
      var protectedSheetCarNumberSheet = sheets[i].protect();
```

```
 //disallow all editors to edit Car Number sheet
 for(var w=0; w<editors.length; w++){
  if(editors[w]!=owner){
   protectedSheetCarNumberSheet.removeEditor(editors[w]);
  }
 }

 var endodometer = "filler";

}else{
 if(sheets[i].getName() == monthToPull){

  Logger.log("The current log is '"+monthToPull+"'");

  //protect the mile allotment cell in the sheet
  var protectedSheetmonthtopull = sheets[i].getRange("E5:E6").protect();
  var protectedScriptThings = sheets[i].getRange("C7:E37").protect();
  //disallow all editors to edit in that cell
  for(var w=0; w<editors.length; w++){
   if(editors[w]!=owner){
    protectedSheetmonthtopull.removeEditor(editors[w]);
    protectedScriptThings.removeEditor(editors[w]);
   }
  }
 }else{

  Logger.log("The current sheet is a previous month");
  //get the odometer for the current sheet
  var endodometer = SpreadsheetApp.openById(logId).getSheetByName(sheets[i].getName()).getRange("A39").getValue();

  //If the sheet is not completed allow editing
  if(endodometer == ""){
   Logger.log("The previous months sheet '"+sheets[i].getName()+"' is not complete - only protecting the mileage allotment");
   //protect the mile allotment cell in the sheet
   var protectedSheetOtherMonth = sheets[i].getRange("E5:E6").protect();

   //disallow all editors to edit in that cell
   for(var w=0; w<editors.length; w++){
    if(editors[w]!=owner){
     protectedSheetOtherMonth.removeEditor(editors[w]);
    }
   }
  }else{
   Logger.log("The previous months sheet '"+sheets[i].getName()+"' is complete - protecting entire sheet");
   //the sheet for a different month is complete - disallow editing
   //protect completed previous months sheet
   var protectedSheetCarNumberSheet = sheets[i].protect();

   //disallow all editors to edit completed previous months sheet
   for(var w=0; w<editors.length; w++){
    if(editors[w]!=owner){
     protectedSheetCarNumberSheet.removeEditor(editors[w]);
    }
   }
  }
 }


}
//hide all sheets except the one we want
```

```
      if(sheets[i].getName()!=monthToPull && endodometer != ""){
        sheets[i].hideSheet();
      }else{
        sheets[i].showSheet();
      }
     }
     Logger.log("Done!")
     break;
    }else{
     Logger.log("Car # "+currentCarLogNumber+ " is not "+carLogToProtect+" - Skipping...")
    }
  }


}


function createNextMonth() {
  //creates the next month's sheet in every car log, also pushes the most updated information assosiated with that log.
  // Set the Active Spreadsheet so we don't forget
  SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
  var master = SpreadsheetApp.getActive();

  //declare the ui for errors
  //var ui = SpreadsheetApp.getUi(); //we don't need it because this is a scheduled task

  //prepare the log template to overwrite new log
  var newLogTemplate = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Log Template");

  //get the month we want to pull
  var monthToPull = master.getRange("A3").getValues() +'-'+ master.getRange("B3").getValues();

  //get the start date for this log
  var startDate = master.getRange("A3").getValues() +'/1/'+ master.getRange("B3").getValues();

  //Get the car log index to find info
  var carLogIndex = SpreadsheetApp.openById(master.getId()).getSheetByName("Master Data").getRange("F:F").getValues();

  //find the folder that contains the actual logs
  var masterId = master.getId();
  var masterFile = DriveApp.getFileById(masterId);
  var masterFolder = masterFile.getParents();
  var childFolder = masterFolder.next().getFolders().next();

  //get a list of mileage logs
  var file = childFolder.getFiles();

  //loop through all mileage logs
  while (file.hasNext()) {//If there is a next file, then continue looping

    //get the file id to pull data
    var currentLog = file.next();
    var logId = currentLog.getId();

    //let's get the car number of the current log
    var carNumber = SpreadsheetApp.openById(logId).getSheetByName("Car Number").getRange("A1").getValue();

    //tell me what log we are on
    //Logger.log(currentLog.getName());

    //get the sheets for the current log
```

```
var sheets = SpreadsheetApp.openById(logId).getSheets();

//true or false variable if the sheet is already made
var sheetAlreadyMade = false;

//loop through all the sheets to check if the sheet we are creating is already made.
for(var i =0;i<sheets.length;i++){
   if (sheets[i].getName()==monthToPull){
     sheetAlreadyMade = true;
   }
}

//if the sheet is not made already, let's make it
if(sheetAlreadyMade == false){

//Copy log template to the new sheet that was also created - do not get rid of SpreadsheetApp.openById(logId)
newLogTemplate.copyTo(SpreadsheetApp.openById(logId));

//rename new sheet to reflect the specified month
SpreadsheetApp.openById(logId).getSheetByName("Copy of Log Template").setName(monthToPull);
}

//the sheet name reflected in monthtoPull should be created now.

//boolian representing whether or not the car was found in "Master Data"
var carFound = true;

//Find what line of the index of "Master Data" the right car is on.
for(var i=0; carLogIndex[i] != carNumber; i++){
   if (i>=carLogIndex.length){
     var carFound = false;
     break;
   }
 }
var index = (i+1);


if(carFound == true){
  //get all the car info to push
  var carInfo = SpreadsheetApp.openById(master.getId()).getSheetByName("Master
Data").getRange("A"+index+":P"+index).getValues();

  //check if there is more than one designated driver
  if(carInfo[0][13]){
    var driver = carInfo[0][12] + ", " + carInfo[0][13];
  }else{
    var driver = carInfo[0][12];
  }

  //check if there is more than one area
  if(carInfo[0][13]){
    var area = carInfo[0][2] + ", " + carInfo[0][3];
  }else{
    var area = carInfo[0][2];
  }

  //Write to the Current Car Log with the newly Updated Information
  var sheetToWriteTo = SpreadsheetApp.openById(logId).getSheetByName(monthToPull);
  sheetToWriteTo.getRange("C2").setValue(carInfo[0][6] +" "+ carInfo[0][7]); //makeModel
  sheetToWriteTo.getRange("C4").setValue(driver); //driver
  sheetToWriteTo.getRange("E5").setValue(carInfo[0][10]); //Allotted Miles
```

```
    sheetToWriteTo.getRange("H2").setValue(carInfo[0][4]); //vin
    sheetToWriteTo.getRange("I3").setValue(carInfo[0][9]); //licensePlate
    sheetToWriteTo.getRange("K2").setValue(startDate); //startDate
    sheetToWriteTo.getRange("K4").setValue(area); //area
    sheetToWriteTo.getRange("I4").setValue(carNumber); //Car Number
   }else{
    Logger.log("Car # " + carNumber + " was not found in 'Master Data', the sheet "+ monthToPull+ " was created, but is empty.
Running 'Push Data' will push the proper information to this car log, so long as the data exists in 'Master Data'.");

    //ui.alert("Car # " + carNumber + " was not found in 'Master Data', the sheet "+ monthToPull+ " was created, but is empty.
Running 'Push Data' will push the proper information to this car log, so long as the data exists in 'Master Data'.");
   }
  }

}


function unprotectAllSheets() {
 //protects all the sheets from editing except those that are incomplete. Run after running either shareWithProperAreaEmails or
shareSingleLogWithProperAreaEmails
 // Set the Active Spreadsheet so we don't forget
 SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
 var master = SpreadsheetApp.getActive();

 //find the folder that contains the actual logs
 var masterId = master.getId();
 var masterFile = DriveApp.getFileById(masterId);
 var masterFolder = masterFile.getParents();
 var childFolder = masterFolder.next().getFolders().next();

 //get a list of mileage logs
 var file = childFolder.getFiles();

 //loop through all mileage logs
 while (file.hasNext()) {//If there is a next file, then continue looping

  //get the file id to pull data
  var currentLog = file.next();
  var logId = currentLog.getId();

  Logger.log(currentLog.getName());

  //get the sheets for the current log
  var sheets = SpreadsheetApp.openById(logId);

  //remove all protections to not create confusion
  var protections = sheets.getProtections(SpreadsheetApp.ProtectionType.SHEET);
  try{
   for (var u = 0; u < protections.length; u++) {
    var protection = protections[u];
    if (protection.canEdit()) {
     protection.remove();
    }
   }
  }
  catch(err){
   Logger.log("no full sheet protections");
  }
  try{
   var protections = sheets.getProtections(SpreadsheetApp.ProtectionType.RANGE);
```

```
      for (var t = 0; t < protections.length; t++) {
        var protection = protections[t];
        if (protection.canEdit()) {
          protection.remove();
        }
      }
    }
    catch(err){
      Logger.log("no range protections");
    }


  }
}


function incompleteMileageLogsEmail () {
  //Emails all the area emails telling them to complete the mileage logs if they are incomplete (for past due mileage logs)

  // Set the Active Spreadsheet so we don't forget
  SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
  var master = SpreadsheetApp.getActive();

  //get the month/sheet we want to interact with
  var monthToPull = master.getRange("A3").getValues() +'-'+ master.getRange("B3").getValues();

  //get the email for the Vehicle report to be sent to
  var emailToSendTo = master.getRange("C3").getValue();

  //declare master data
  var masterData = SpreadsheetApp.openById(master.getId()).getSheetByName("Master Data");

  //Get the car log index to find info
  var carLogIndex = masterData.getRange("F:F").getValues();

  //Get the are email index
  var areaEmailIndex = masterData.getRange("O:P").getValues();

  //find the folder that contains the actual logs
  var masterId = master.getId();
  var masterFile = DriveApp.getFileById(masterId);
  var masterFolder = masterFile.getParents();
  var childFolder = masterFolder.next().getFolders().next();

  //get a list of mileage logs
  var file = childFolder.getFiles();

  //loop through all mileage logs
  while (file.hasNext()) {//If there is a next file, then continue looping

    //get the file id to pull data
    var currentLog = file.next();
    var logId = currentLog.getId();

    Logger.log(currentLog.getName());

    //check to see if the log sheet we want exists
    var logSheets = SpreadsheetApp.openById(logId).getSheets();

    //boolian var to represent if log sheet exists
    var sheetExists = false;
```

```
  //loop through all sheets to make sure MonthToPull exists in the log
  for(var w = 0; w<logSheets.length; w++){
   if(logSheets[w].getName()==monthToPull){
    var sheetExists = true;
    break;
   }
  }

  //let's do stuff if it exists
  if(sheetExists == true){

   //check if the sheet is complete (whether or not we need to email them)
   var endOdimeter = SpreadsheetApp.openById(logId).getSheetByName(monthToPull).getRange("A39").getValue();

   //if the sheet isn't complete send an email
   if(endOdimeter == ""){
    //Find Current Car log we are working with
    var carNumber = SpreadsheetApp.openById(logId).getSheetByName("Car Number").getRange("A1").getValue();

    //find that car log in the master data index
    for(var e=0; carLogIndex[e] != carNumber; e++){
     if (e>=carLogIndex.length){
      Logger.log("The car wasn't found in the index... check master data");
      break;
     }
    }
    var currentCarInfoLine = e;

    //Logger.log(areaEmailIndex[currentCarInfoLine][0]);
    // Set the text body to attach to the email.
    var body = "Your mileage log '"+ currentLog.getName()+"' is incomplete. "+"Please finish filling sheet '"+ monthToPull +"' in!
"+ currentLog.getUrl();

    // Construct the Subject Line
    var subject = "Finish Your OVERDUE Car Log!";

    //Time to send them an email
    MailApp.sendEmail(areaEmailIndex[currentCarInfoLine][0], subject, body);

    //see if the car is shared
    if(areaEmailIndex[currentCarInfoLine][1]!=""){
     MailApp.sendEmail(areaEmailIndex[currentCarInfoLine][1], subject, body);
    }

    //create a report to send to Vehicle cordinator
    var fleetReport = [fleetReport, currentLog.getName()+"'s - '"+monthToPull+"' is INCOMPLETE - Status: Email Sent."];

   }else{
    Logger.log("The mileage log '" +currentLog.getName()+"' is complete!");
   }
  }else{
   Logger.log("Sheet '"+monthToPull+"' does not exist in '"+currentLog.getName()+"'!")
  }
 }
 MailApp.sendEmail(emailToSendTo, "Driver Logs that are incomplete", fleetReport.toString().replaceAll(',', '\n'));
}


function finishMileageLogsReminderEmail () {
 //Emails all the area emails telling them to complete the mileage logs if they are incomplete on the 1st of the month
```

```
// Set the Active Spreadsheet so we don't forget
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
var master = SpreadsheetApp.getActive();

//get the month/sheet we want to interact with
var monthToPull = master.getRange("A3").getValues() +'-'+ master.getRange("B3").getValues();

//get the email to send the report to
var emailToSendTo = master.getRange("C3:D3").getValue();

//declare master data
var masterData = SpreadsheetApp.openById(master.getId()).getSheetByName("Master Data");

//Get the car log index to find info
var carLogIndex = masterData.getRange("F:F").getValues();

//Get the are email index
var areaEmailIndex = masterData.getRange("O:P").getValues();

//find the folder that contains the actual logs
var masterId = master.getId();
var masterFile = DriveApp.getFileById(masterId);
var masterFolder = masterFile.getParents();
var childFolder = masterFolder.next().getFolders().next();

//get a list of mileage logs
var file = childFolder.getFiles();

//var for report at the end
var fleetReport = "";

//loop through all mileage logs
while (file.hasNext()) {//If there is a next file, then continue looping

 //get the file id to pull data
 var currentLog = file.next();
 var logId = currentLog.getId();

 Logger.log("Checking "+currentLog.getName());

 //check to see if the log sheet we want exists
 var logSheets = SpreadsheetApp.openById(logId).getSheets();

 //boolian var to represent if log sheet exists
 var sheetExists = false;

 //loop through all sheets to make sure MonthToPull exists in the log
 for(var w = 0; w<logSheets.length; w++){
  if(logSheets[w].getName()==monthToPull){
   var sheetExists = true;
   break;
  }
 }

 //let's do stuff if it exists
 if(sheetExists == true){

  //check if the sheet is complete (whether or not we need to email them)
  var endOdimeter = SpreadsheetApp.openById(logId).getSheetByName(monthToPull).getRange("A39").getValue();
```

```
    //if the sheet isn't complete send an email
    if(endOdimeter == ""){
     //Find Current Car log we are working with
     var carNumber = SpreadsheetApp.openById(logId).getSheetByName("Car Number").getRange("A1").getValue();

     //find that car log in the master data index
     for(var e=0; carLogIndex[e] != carNumber; e++){
      if (e>=carLogIndex.length){
       Logger.log("The car wasn't found in the index... check master data");
       break;
      }
     }
     var currentCarInfoLine = e;

     //Logger.log(areaEmailIndex[currentCarInfoLine][0]);
     // Set the text body to attach to the email.
     var body = "Make sure to finish your mileage log before the second of the month! Add all receipts and the end odometer! Do Not fill in the end odometer until you are completely done!"+ currentLog.getUrl();

     // Construct the Subject Line
     var subject = "Finish Your Car Log Reminder!";

     //Time to send them an email
     //MailApp.sendEmail(areaEmailIndex[currentCarInfoLine][0], subject, body);
     var fleetReport = [fleetReport, currentLog.getName()];

     //see if the car is shared
     if(areaEmailIndex[currentCarInfoLine][1]!=""){
      //MailApp.sendEmail(areaEmailIndex[currentCarInfoLine][1], subject, body);
     }

    }else{
     Logger.log("The mileage log '" +currentLog.getName()+"' is complete!");
    }
   }else{
    Logger.log("Sheet '"+monthToPull+"' does not exist in '"+currentLog.getName()+"'!")
   }
  }
  var report = fleetReport.toString();
  var report = report.replaceAll(',', '\n');
  MailApp.sendEmail(emailToSendTo, "Finish mileage Logs reminder report","An email was sent to remind missionaries to finish their car logs.\nEmail sent for:\n"+ report);
}


function updateMonthToCurrentMonth (){
  // Set the Active Spreadsheet so we don't forget
  SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
  var master = SpreadsheetApp.getActive();

  //get the current date
  var d = new Date();
  var month = d.getMonth();
  month++;
  var year = d.getFullYear();

  if (month >= 10){
   master.getSheetByName("Script Master Sheet").getRange("A3").setValue(month);
   master.getSheetByName("Script Master Sheet").getRange("B3").setValue(year);
  }else{
   master.getSheetByName("Script Master Sheet").getRange("A3").setValue("0"+month);
```

```
    master.getSheetByName("Script Master Sheet").getRange("B3").setValue(year);
  }


}


function updateMonthToPreviousMonth (){
  // Set the Active Spreadsheet so we don't forget
  SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Script Master Sheet").activate();
  var master = SpreadsheetApp.getActive();

  //get the current date
  var d = new Date();
  var month = d.getMonth();  //if January - January = 0
  var year = d.getFullYear();
  if (month == 0){
    var month = 12;
    year--;
  }


  if (month >= 10){
    master.getSheetByName("Script Master Sheet").getRange("A3").setValue(month);
    master.getSheetByName("Script Master Sheet").getRange("B3").setValue(year);
  }else{
    master.getSheetByName("Script Master Sheet").getRange("A3").setValue("0"+month);
    master.getSheetByName("Script Master Sheet").getRange("B3").setValue(year);
  }

}


// Trigger Functions
function triggerCreateNextMonth(){
  try{
    updateMonthToCurrentMonth ();
    createNextMonth(); //FIRST TIME
  }
  catch(err){
    Logger.log("WE HAD AN ERROR ONCE, TRYING TO RUN createNextMonth() AGAIN!");
    try{
      updateMonthToCurrentMonth ();
      createNextMonth(); //SECOND TIME
    }
    catch(err){
      Logger.log("WE HAD AN ERROR TWICE, TRYING TO RUN createNextMonth() AGAIN!");
      try{
        updateMonthToCurrentMonth ();
        createNextMonth(); //THIRD TIME
      }
      catch(err){
        Logger.log("WE HAD AN ERROR THREE TIMES, RUN THE THING MANUALLY");
      }
    }
  }
}

function triggerincompleteMileageLogsEmail(){
  try{
    updateMonthToPreviousMonth ();
```

```
    incompleteMileageLogsEmail(); //FIRST TIME
   }
   catch(err){
    Logger.log("WE HAD AN ERROR ONCE, TRYING TO RUN incompleteMileageLogsEmail() AGAIN!");
    try{
     updateMonthToPreviousMonth ();
     incompleteMileageLogsEmail(); //SECOND TIME
    }
    catch(err){
     Logger.log("WE HAD AN ERROR TWICE, TRYING TO RUN incompleteMileageLogsEmail() AGAIN!");
     try{
      updateMonthToPreviousMonth ();
      incompleteMileageLogsEmail(); //THIRD TIME
     }
     catch(err){
      Logger.log("WE HAD AN ERROR THREE TIMES, RUN THE THING MANUALLY");
     }
    }
   }
}

function triggerprotectSheets(){
 try{
  updateMonthToCurrentMonth();
  protectSheets(); //FIRST TIME
 }
 catch(err){
  Logger.log("WE HAD AN ERROR ONCE, TRYING TO RUN protectSheets() AGAIN!");
  try{
   updateMonthToCurrentMonth();
   protectSheets(); //SECOND TIME
  }
  catch(err){
   Logger.log("WE HAD AN ERROR TWICE, TRYING TO RUN protectSheets() AGAIN!");
   try{
    updateMonthToCurrentMonth();
    protectSheets(); //THIRD TIME
   }
   catch(err){
    Logger.log("WE HAD AN ERROR THREE TIMES, RUN THE THING MANUALLY");
   }
  }
 }
}

function triggerfinishMileageLogsReminderEmail(){
 try{
  updateMonthToPreviousMonth ();
  finishMileageLogsReminderEmail(); //FIRST TIME
  updateMonthToCurrentMonth();
 }
 catch(err){
  Logger.log("WE HAD AN ERROR ONCE, TRYING TO RUN finishMileageLogsReminderEmail() AGAIN!");
  try{
   updateMonthToPreviousMonth ();
   finishMileageLogsReminderEmail(); //SECOND TIME
   updateMonthToCurrentMonth();
  }
  catch(err){
   Logger.log("WE HAD AN ERROR TWICE, TRYING TO RUN finishMileageLogsReminderEmail() AGAIN!");
   try{
```

```
      updateMonthToPreviousMonth ();
      finishMileageLogsReminderEmail(); //THIRD TIME
      updateMonthToCurrentMonth();
     }
     catch(err){
      Logger.log("WE HAD AN ERROR THREE TIMES, RUN THE THING MANUALLY");
     }
    }
  }
}

function triggeremailAllReportsLandscape(){
  try{
    updateMonthToPreviousMonth ();
    emailAllReportsLandscape(); //first TIME
    updateMonthToCurrentMonth();
  }
  catch(err){
    Logger.log("WE ONLY WANT TO RUN THIS ONCE TO AVOID SPAM, SO RUN THE THING MANUALLY or check each
mileage log separately!");
  }
}
```

Part 2 – Excel Macro for converting Vehicles Assignment report

This source code can be found in the “Convert Vehicles Assignment report for VMLC.bas” file.

```
Sub Convert_Vehicles_Assignment_Report_For_VMLC()
'
' ConvertVehiclesAssignemntReportForVMLC Macro
'

'
    Range("A1").Select
    ActiveCell.Rows("1:2").EntireRow.Select
    Selection.Delete Shift:=xlUp
    Selection.End(xlDown).Select
    ActiveCell.Offset(1, 0).Range("A1:L4").Select
    Selection.EntireRow.Delete

    Columns("D:L").Select
    Selection.Copy
    Columns("E:E").Select
    ActiveSheet.Paste
    Columns("D:D").Select
    Selection.ClearContents
    ActiveWindow.ScrollColumn = 3
    ActiveWindow.ScrollColumn = 2
    ActiveWindow.ScrollColumn = 1
    Range("C1").Select
    Selection.Copy
    Range("D1").Select
    ActiveSheet.Paste
    Application.CutCopyMode = False
    ActiveCell.FormulaR1C1 = "Area 2"
```

```
Range("D2").Select
ActiveWindow.ScrollColumn = 2
ActiveWindow.ScrollColumn = 3
ActiveWindow.ScrollColumn = 4
ActiveWindow.ScrollColumn = 5
Range("M1").Select
ActiveCell.FormulaR1C1 = "Designated Driver"
Range("M1").Select
Selection.Copy
Range("N1").Select
ActiveSheet.Paste
ActiveCell.FormulaR1C1 = "Designated Driver 2"
Cells.Replace what:="" & Chr(10) & "", Replacement:="", LookAt:=xlPart, SearchOrder:= _
    xlByRows, MatchCase:=False, SearchFormat:=False, ReplaceFormat:=False, _
    FormulaVersion:=xlReplaceFormula2
Cells.Replace what:=" - Can Drive - Designated Driver", Replacement:="_", _
    LookAt:=xlPart, SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:= _
    False, ReplaceFormat:=False, FormulaVersion:=xlReplaceFormula2
Cells.Replace what:="_*", Replacement:="", LookAt:=xlPart, SearchOrder _
    :=xlByRows, MatchCase:=False, SearchFormat:=False, ReplaceFormat:=False _
    , FormulaVersion:=xlReplaceFormula2
Cells.Replace what:="Cannot", Replacement:="Can", LookAt:=xlPart, _
    SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
    ReplaceFormat:=False, FormulaVersion:=xlReplaceFormula2
Cells.Replace what:="*Can Drive", Replacement:="", LookAt:=xlPart, _
    SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
    ReplaceFormat:=False, FormulaVersion:=xlReplaceFormula2
Columns("M:M").ColumnWidth = 24.86
Columns("M:N").Select
Selection.ColumnWidth = 25.14
Columns("K:K").Select
Selection.ColumnWidth = 15.29
Columns("I:I").Select
Selection.ColumnWidth = 20.29
Columns("H:H").ColumnWidth = 13.86
Columns("G:G").Select
Selection.ColumnWidth = 12.57
ActiveWindow.ScrollColumn = 4
ActiveWindow.ScrollColumn = 3
ActiveWindow.ScrollColumn = 2
ActiveWindow.ScrollColumn = 1
Columns("C:D").Select
Range("D1").Activate
Selection.ColumnWidth = 10
Columns("A:B").Select
Range("B1").Activate
Selection.ColumnWidth = 12.57
Range("A1").Select

Columns("F:F").Select
Selection.FormatConditions.AddUniqueValues
Selection.FormatConditions(Selection.FormatConditions.Count).SetFirstPriority
Selection.FormatConditions(1).DupeUnique = xlDuplicate
With Selection.FormatConditions(1).Font
    .Color = -16383844
    .TintAndShade = 0
End With
```

```
With Selection.FormatConditions(1).Interior
    .PatternColorIndex = xlAutomatic
    .Color = 13551615
    .TintAndShade = 0
End With
Selection.FormatConditions(1).StopIfTrue = False
Application.CutCopyMode = False
ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Clear
ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Add2 Key:=Range("F1:F83") _
    , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
With ActiveWorkbook.Worksheets("Sheet1").Sort
    .SetRange Range("A2:N83")
    .Header = xlNo
    .MatchCase = False
    .Orientation = xlTopToBottom
    .SortMethod = xlPinYin
    .Apply
End With
Range("F1").Select
ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Clear
ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Add(Range("F2:F79"), _
    xlSortOnFontColor, xlAscending, , xlSortNormal).SortOnValue.Color = RGB(156, 0 _
    , 6)
With ActiveWorkbook.Worksheets("Sheet1").Sort
    .SetRange Range("A1:N79")
    .Header = xlYes
    .MatchCase = False
    .Orientation = xlTopToBottom
    .SortMethod = xlPinYin
    .Apply
End With


fTwo = Range("F2").value
fThree = Range("F3").value
Do While fTwo = fThree


    Range("C3").Select
    Selection.Cut
    Range("D2").Select
    ActiveSheet.Paste
    Range("M3").Select
    Selection.Cut
    Range("N2").Select
    ActiveSheet.Paste
    Range("K2").Select
    Selection.Copy
    Range("J3").Select
    ActiveSheet.Paste
    Range("I3").Select
    Application.CutCopyMode = False
    Application.CutCopyMode = False
    ActiveCell.FormulaR1C1 = "=RC[1]+RC[2]"
    Range("I3").Select
    Selection.Copy
    Range("K2").Select
```

```
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Rows("3:3").Select
    Application.CutCopyMode = False
    Selection.Delete Shift:=xlUp
    Range("F1").Select


    ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Add(Range("F2:F78"), _
        xlSortOnFontColor, xlAscending, , xlSortNormal).SortOnValue.Color = RGB(156, 0 _
        , 6)
    With ActiveWorkbook.Worksheets("Sheet1").Sort
        .SetRange Range("A1:N78")
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With


    fTwo = Range("F2").value
    fThree = Range("F3").value

  Loop

  Columns("F:F").Select
  ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Clear
  ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Add2 Key:=Range("F1:F83") _
      , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
  With ActiveWorkbook.Worksheets("Sheet1").Sort
      .SetRange Range("A1:N83")
      .Header = xlYes
      .MatchCase = False
      .Orientation = xlTopToBottom
      .SortMethod = xlPinYin
      .Apply
  End With
  Columns("O:IV").Select
  Range("O26").Activate
  Selection.EntireColumn.Hidden = True
  Range("A1").Select

End Sub
```

Part 3 – Excel Macro for getting the list of area emails

This source code can be found in the “Area_Email_From_Roster” file.

```
Sub Area_Email_From_Roster()
'
' Area_Email_From_Roster Macro
'

'
    Rows("1:3").Select
    Selection.Delete Shift:=xlUp
    Selection.End(xlDown).Select
    ActiveCell.Offset(1, 0).Range("A1:AA2").Select
    Selection.EntireRow.Delete
    ActiveWindow.SmallScroll Down:=-300
    Range("A1").Select

    Columns("A:M").Select
    Selection.Delete Shift:=xlToLeft
    Columns("C:F").Select
    Selection.Delete Shift:=xlToLeft
    Columns("A:B").Select
    Selection.ColumnWidth = 26.14

    Columns("A:B").Select
    ActiveSheet.Range("$A$1:$B$226").removeduplicates Columns:=Array(1, 2), _
       Header:=xlYes
    Columns("A:A").Select
    Selection.Cut
    Columns("C:C").Select
    ActiveSheet.Paste
    Columns("A:A").Select
    Selection.Delete Shift:=xlToLeft

    Rows("2:2").Select
    Selection.Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
    Range("A2").Select
    ActiveCell.FormulaR1C1 = "office"
    Columns("C:IV").Select
    Selection.EntireColumn.Hidden = True
    Range("A1").Select
End Sub
```

Chapter 8 - How to setup this Project for your Missions use

Part 1 – Guide for setup

So you decided to mooch of my hard work?

Well I guess I kind of wanted you to do that since I included this section.

Basically there are three steps to getting this project set up for your benefit.

1. **Getting a copy of everything you need in the right place.**
2. **Setting up the automated tasks**
3. **Maintenance**

This is the directory tree of this project

My Drive

|--------Mileage Reports

| |--------Virtual Mileage Logs Controller (VMLC)

| |--------Area_Email_From_Roster.bas

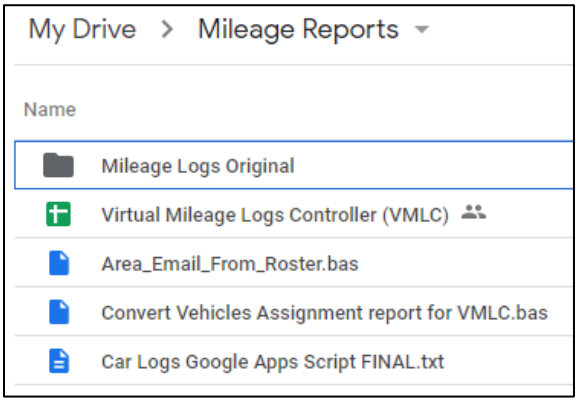
| |--------Convert Vehicles Assignment report for VMLC.bas

| |--------Car Logs Google Apps Script FINAL.txt

| |--------Mileage Logs Original

| |--------(The actual mileage logs will be in this folder)

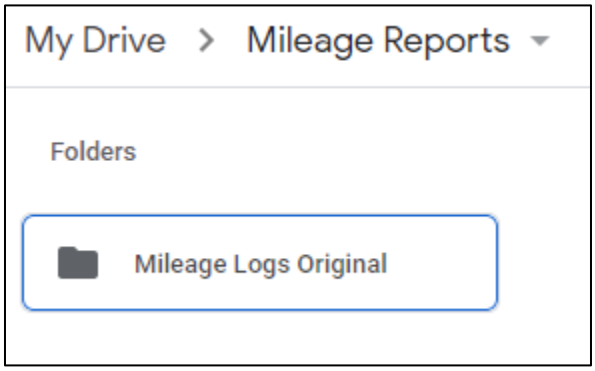
If this doesn't make sense here is a picture.



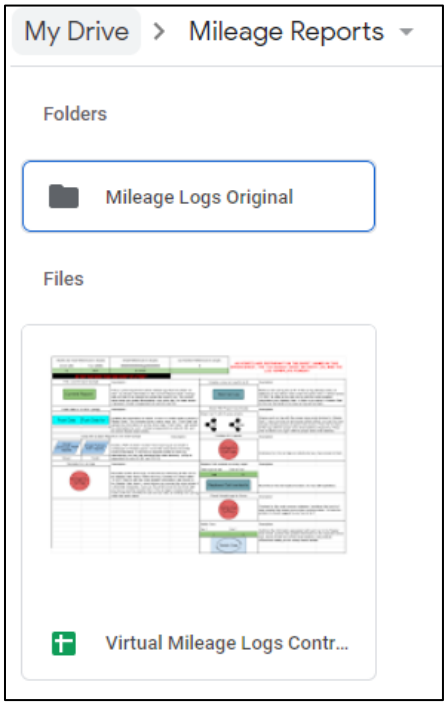
You will have to generate this same directory tree yourself (cuz duplicating folders isn't really a thing in Google Drive). I will give you the files you will need after I instruct you how to build your own directory tree.

Getting a copy of everything you need in the right place.

1. Go to your Google Drive.
2. Create a folder called "Mileage Reports".
3. Inside of the "Mileage Reports" folder create another folder, this time named "Mileage Logs Original". It should look something like this.



4. Now you are ready for files.
5. Duplicate this file into your "Mileage Reports" folder (this is the Virtual Mileage Logs Controller (VMLC)). The URL to this file is "https://docs.google.com/spreadsheets/d/1Dm7MITACGn1jvmVFh8m0wOLQxjzGlvfJrNDJ-SYU7sE/edit?usp=sharing". It should look something like this. *I will not accept ANY edit access requests, so don't waste my time or yours.*

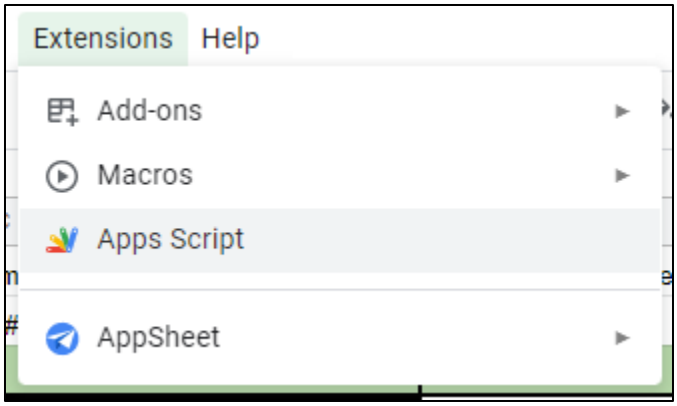


6. If you duplicate this file correctly the background functions and their assigned buttons should be intact. If you downloaded and then uploaded this file the assigned buttons and functions will not be present. (in other words try again) (If you wish to download the source code, the file containing the google apps script source code is located here. (it is in txt format) (URL is "https://drive.google.com/file/d/1H1sOCYlZDdsmMcaxWv7iJusXWYzBYAMw/view?usp=sharing"))
7. We also want access to the excel macro files so we can utilize them.
8. For the "Area_Email_From_Roster.bas" file the url is below or click here "https://drive.google.com/file/d/1jSN1hd8x3PSlgyL9OpbbNBsGGwU33GTC/view?usp=sharing".
9. For the "Convert Vehicles Assignment report for VMLC.bas" file the URL is below or click here "https://drive.google.com/file/d/1iDhOgBHDrEi-t9s_JmdwkiO2-gC_zHPQ/view?usp=sharing".
10. It would be recommended to put these files in the same directory as the "Virtual Mileage Logs Controller (VMLC)" file so that during office staff transitions they don't have to these steps again.
11. Download both of these files to your computer.
12. As a note of interest, when downloading any type of .bas file chrome and other web browsers will warn that the file may be dangerous. So if you think that either one of these files/macros are a virus, (they are not I recorded them) then you are welcome to not use them or use the documented source code above to create them yourself.
13. You will then need to import each of these Excel macros into your personal workbook (follow Chapter 3-part 4).
14. Then update the information in the "Virtual Mileage Logs Controller (VMLC)" specifically the "Master Data" and "List of all area emails" sheet. (Chapter 3) After this, create all the driver logs by using the "Recreate ALL car logs" button.
15. Once this is complete you have all the files in place to move on to creating the triggers (automated functions) for this project.

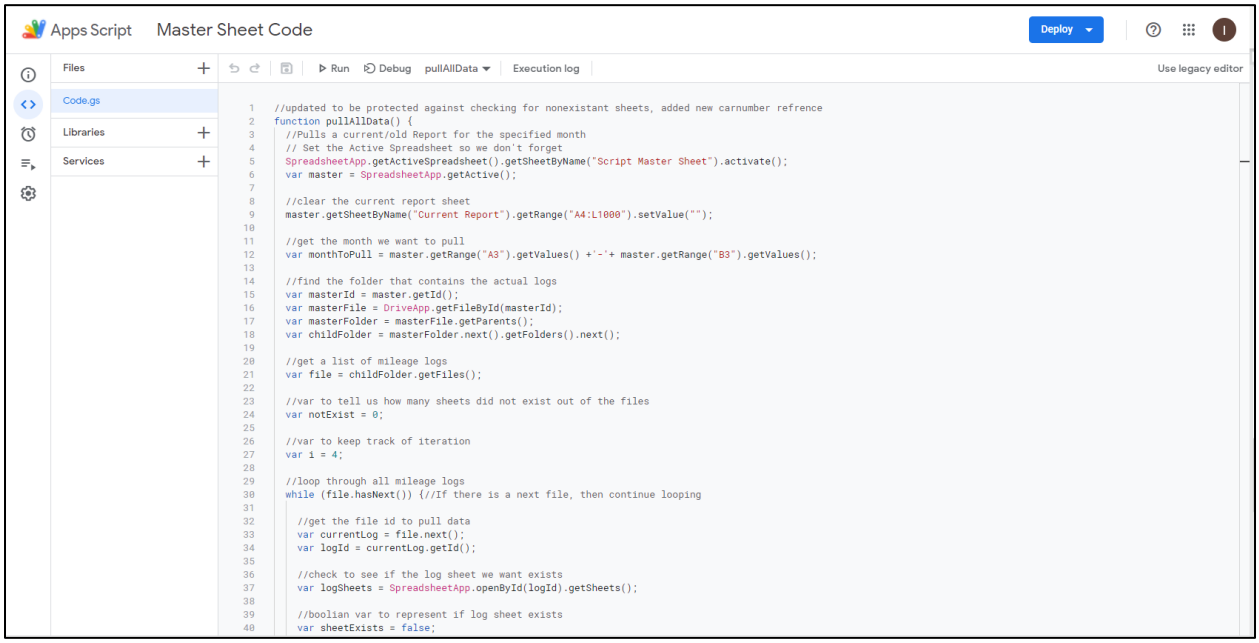
Setting up the automated tasks

At this point in the project all the driver logs should be created, the information up to date, and the directory tree intact.

1. Open the "Virtual Mileage Logs Controller (VMLC)".
2. Click on the "Extensions" then "Apps Script".

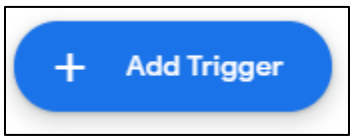


3. A new tab will open showing you the google apps script code. It looks something like this.

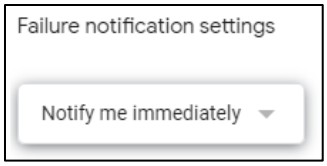


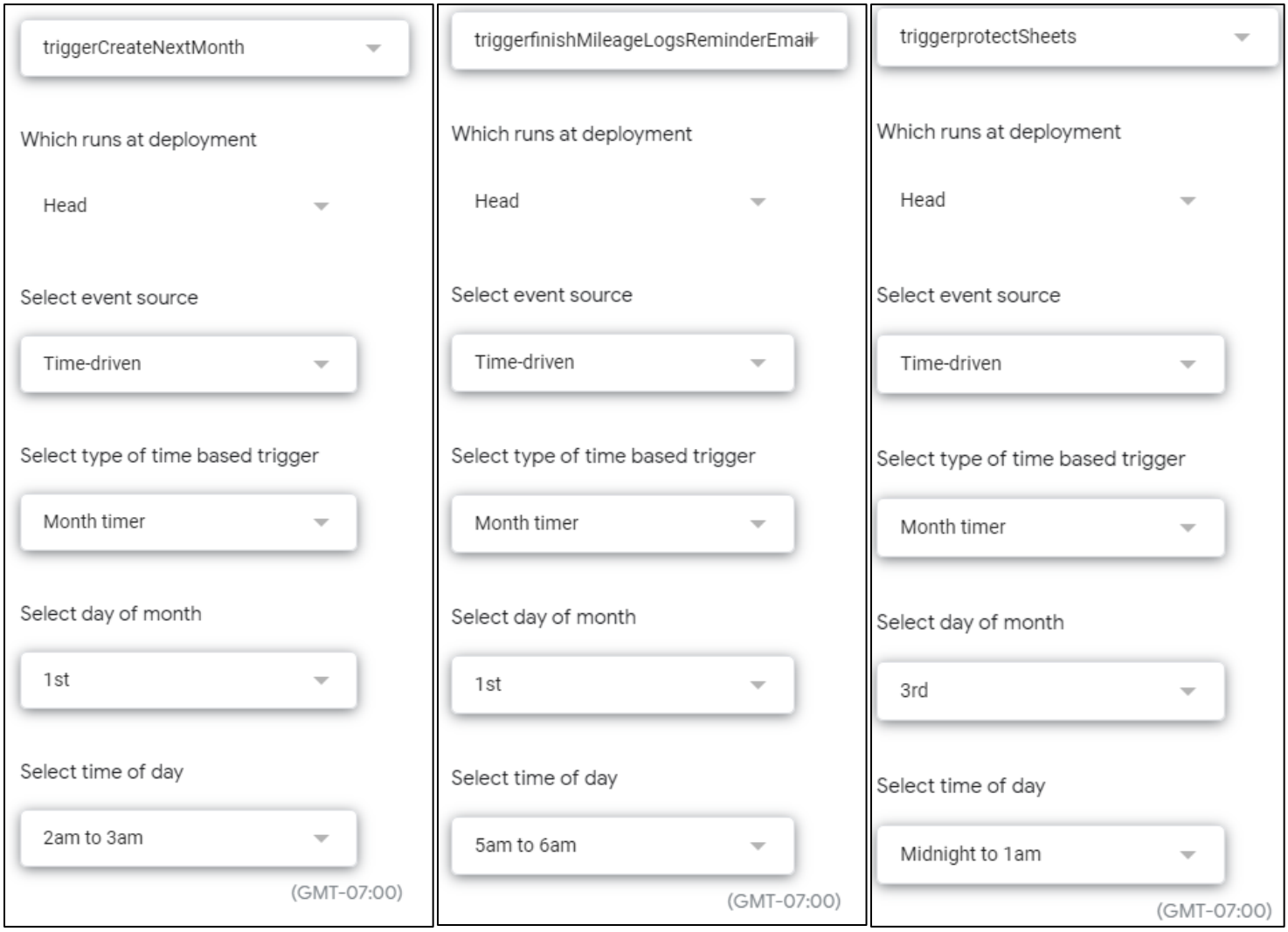
4. Click on the alarm icon on the left panel of the screen (triggers).

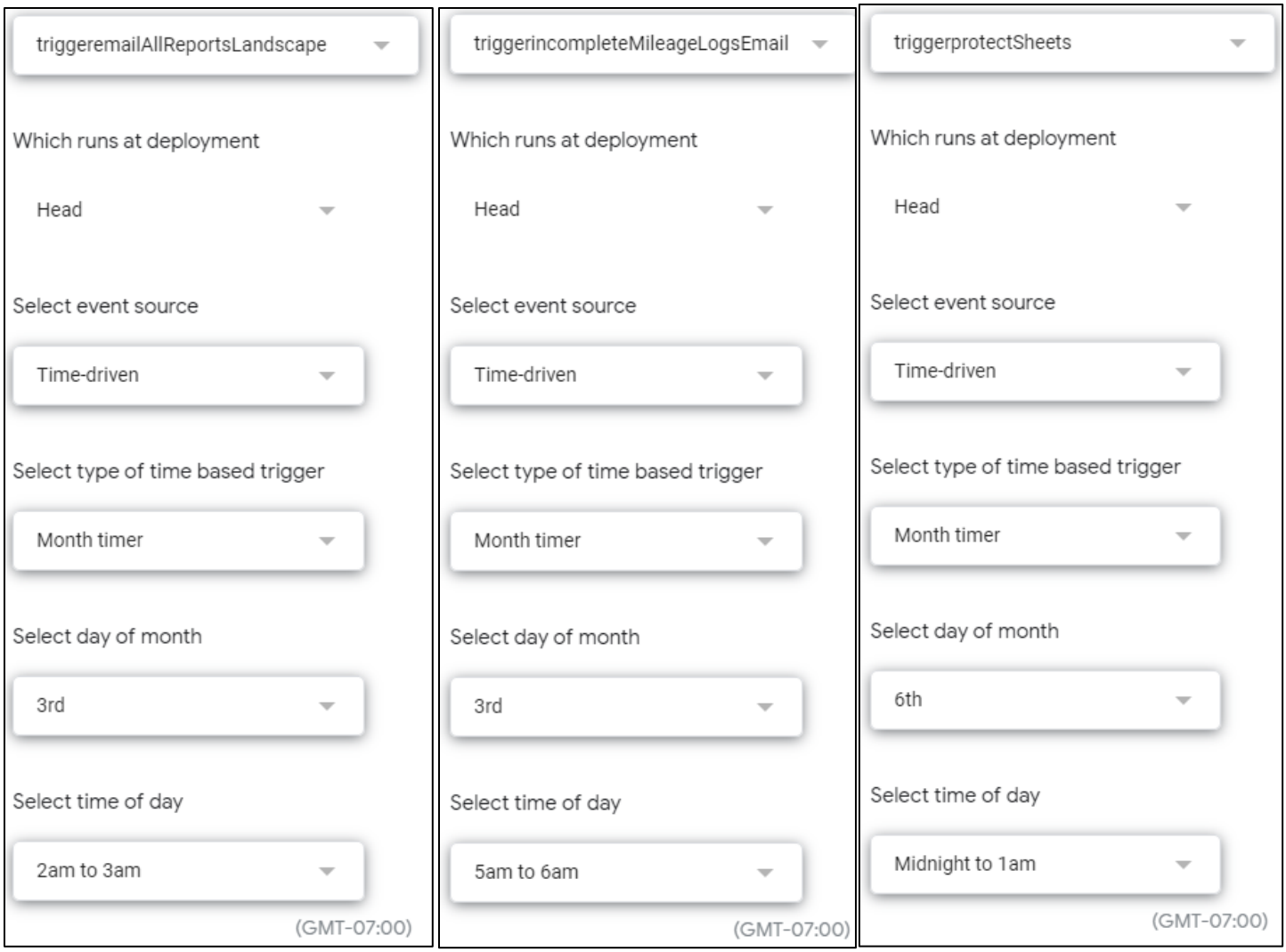
5. You will be creating 6 triggers.
6. Click the “Add Trigger” button.



7. Create each trigger with the settings as follows:
8. Each trigger will be “Failure Notification Settings” will be set to “Notify me Immediately”




triggerCreateNextMonth
Which runs at deployment
Head
Select event source
Time-driven
Select type of time based trigger
Month timer
Select day of month
1st
Select time of day
2am to 3am
(GMT-07:00)
triggerfinishMileageLogsReminderEmail
Which runs at deployment
Head
Select event source
Time-driven
Select type of time based trigger
Month timer
Select day of month
1st
Select time of day
5am to 6am
(GMT-07:00)
triggerprotectSheets
Which runs at deployment
Head
Select event source
Time-driven
Select type of time based trigger
Month timer
Select day of month
3rd
Select time of day
Midnight to 1am
(GMT-07:00)



Maintenance

Pretty much what is covered in Chapter 3.

Congrats you made it to the end!

Those at Salt Lake who actually read this, Good Job! You are given permission from me (Elder Colby James Ransom) to use this manual and source code as building blocks to create something for the missions who are stuck on the physical logs or any type of driver log – just be sure to give credit to God and to me for submitting this project.