

Polisy Data Collection

Contents

| | |
|--|---|
| Recording the Data | 2 |
| Reporting the Data..... | 3 |
| Getting and Storing the Email Data | 4 |
| Parsing the Email Data | 5 |
| Graphing and Using the Email Data | 7 |

Sometimes it is necessary to record data points in your home controller system in order to determine the best way to resolve something. For example, you might want to track your air conditioner usage over time to see when it was being used most so that you change when it turns on in order to save electricity.

In this document I will describe the steps that I took to accomplish this goal. Although I use specific thermostat references, it should be clear that this approach can easily be adjusted to any sort of data collection.

I use the TimeData and Climacell (weather) nodes in this example as well as a Trane Z-Wave thermostat. I also use Thunderbird as an email client – if you use another client, your data may vary slightly.

Note in this example, I describe the cooling data however there are several other data points that are also being collected.

Recording the Data

A simple routine to record how long the A/C ran can be seen below. This accumulates the total run time in the variable \$CoolMinutesTotalMain.

```
Program Content for 'Track Main Cooling'
If
    'Thermostat Main Area' Heat/Cool State is Cooling
Then
    $CoolStartTimeMain = 'TimeData' Minutes YTD
Else
    $CoolMinutesMain = 'TimeData' Minutes YTD
    $CoolMinutesMain -= $CoolStartTimeMain
    $CoolMinutesTotalMain += $CoolMinutesMain
    $CoolStartTimeMain = 'TimeData' Minutes YTD

idle -> cooling -> pending cooling -> idle
```

Reporting the Data

I was not able to find a way to store arrays of values in Polisy. The way I chose to accomplish this was using Polisy's custom email notifications. By recording the data and then sending it to myself via email once per hour I am able to collect 24 data points every day. The logic to do so is shown below.

```
Program Content for 'Report Cooling Stats'
If
  'TimeData' Minute is 0 minute
Then
  $PeakOutsideTemperature = 'Climate11 Weather' Temperature 'F'
  $ACBedMinutes = $CoolMinutesTotalBeds
  $ACMainMinutes = $CoolMinutesTotalMain
  $ACBedSeconds = $DoorBedOpenSeconds
  $ACFrontSeconds = $DoorFrontOpenSeconds
  $ACGarageSeconds = $DoorGarageOpenSeconds
  $ACSliderSeconds = $DoorSliderOpenSeconds
  $ACWindowsSeconds = $WindowsOpenSeconds
  $CoolMinutesTotalBeds = 0
  $CoolMinutesTotalMain = 0
  $DoorBedOpenSeconds = 0
  $DoorFrontOpenSeconds = 0
  $DoorGarageOpenSeconds = 0
  $DoorSliderOpenSeconds = 0
  $WindowsOpenSeconds = 0
  Send Notification to 'Alec and Marilyn' content 'Cooling Stats'
Else
  - No Actions - (To add one, press 'Action')
```

Note that Polisy does not protect the variables used in Notifications so you have to copy the data to a temporary variable if it needs to be zeroed out as above. In this case the variables that start with “\$AC...” are the temporary variables used by the notification, whereas the others are actual data collection points.

The ‘Cooling Stats’ notification is a simple email that sends the cooling data points (stored in the temporary variables) to my email account.

Custom Notification 'Cooling Stats' [5]

From:

Subject: Cooling Stats for \${sys.date}

Body: Bedroom A/C ran for \${var.1.19} minutes
Main A/C ran for \${var.1.18} minutes

Peak outside temperature was \${var.1.17} F

Garage Door open for \${var.1.30} seconds
Front Door open for \${var.1.31} seconds
Slider Door open for \${var.1.29} seconds
Bed Door open for \${var.1.28} seconds
Windows open for \${var.1.33} seconds

Text HTML XML

Alert Default Subject Add Variable

Ok Cancel

Getting and Storing the Email Data

In my email client I have created a filter that picks out the emails based upon seeing the words “Cooling Stats” in the subject line and places them in a specific folder.

I then select all of the Cooling Stats emails in the folder and use my email clients “Save As” function to save them as individual files on my hard drive. The data from these emails includes lots of header information that is not needed. Shown below is an example email file.

```
X-Mozilla-Status: 0001
X-Mozilla-Status2: 00000000
X-Mozilla-Keys:
Delivered-To: *****@gmail.com
Received: by 2002:a05:6520:4d01:b0:1be:f098:e203 with SMTP id cy1csp2410727lkb;
    Mon, 16 May 2022 10:00:30 -0700 (PDT)
```

```
**** LOTS OF HEADER DATA REMOVED FOR CLARITY ****
```

```
Date: Mon, 16 May 2022 10:00:29 -0700
Subject: Cooling Stats for 2022/05/16
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8
X-Antivirus: AVG (VPS 220516-8, 5/16/2022), Inbound message
X-Antivirus-Status: Clean
```

```
Bedroom A/C ran for 14 minutes
Main A/C ran for 14 minutes
```

```
Peak outside temperature was 85 F
```

As can be seen, this data needs to be parsed before it can be of any practical use.

Parsing the Email Data

A simple program can be written to parse the data and extract the needed data in comma separated format that is usable in Excel. I chose to use C#, but it could be written in any language. My program is shown below for those who wish to use C#. This parses all of the *.eml files (generated by Thunderbird in the previous step) stripping out the required information and stores the data from each file as a line in the CoolingStats.txt output file that is generated in the same folder.

```
//#define CONSOLE_OUTPUT

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace CoolingStats
{
    class Program
    {
        static void Main(string[] args)
        {
            var directory = @"C:\Users\Alec\Desktop\Cooling Stats";
            var outFileName = directory + @"\CoolingStats.txt";

            using (StreamWriter outFile = new StreamWriter(outFileName))
            {
                string[] allFiles = Directory.GetFiles(directory, "*.eml");
                foreach (string fileName in allFiles)
                {
                    Console.WriteLine(fileName);
                    IEnumerable<string> lines = File.ReadLines(fileName);

                    string CSVOutput = "";
                    int prefixesFound = 0;
                    string timeString = "XX:XX", dateString = "XX/XX/XXXX";

                    foreach (string line in lines)
                    {
                        string[] prefixes = new string[] { "Bedroom A/C ran for ", "Main A/C ran for ",
"Peak outside temperature was ", "Garage Door open for ", "Front Door open for ", "Slider Door open for ",
"Bed Door open for ", "Windows open for " };

                        #if CONSOLE_OUTPUT
                            Console.WriteLine("----> " + line);
                        #endif

                        if (line.StartsWith("Date: "))
                        {
                            #if CONSOLE_OUTPUT
                                Console.WriteLine("+++ " + line.Substring(6));
                            #endif

                            timeString = line.Substring(line.IndexOf(':', 6) - 2, 5);
                        }
                        else
                        {
                            var subject = "Subject: Cooling Stats for ";

                            if (line.StartsWith(subject))
                            {
                                dateString = line.Substring(subject.Length);

                                CSVOutput = dateString + "," + timeString;
                            }
                            else

```

```

        {
            foreach (string prefix in prefixes)
            {
                if (line.Contains(prefix))
                {
                    int length = line.IndexOf(' ', prefix.Length) - prefix.Length;
                    var valueString = line.Substring(prefix.Length, length);
                    int value = int.Parse(valueString);

                    Console.WriteLine("+++ " + length + ", " + valueString + ", " +
value);
                    #endif

                    CSVOutput += "," + valueString;
                    prefixesFound++;
                }
            }
        }

        // add missing fields to earlier records with zero values
        while (prefixesFound < 8)
        {
            CSVOutput += ",0";
            prefixesFound++;
        }

        outFile.WriteLine(CSVOutput);
    }
}
#endif
Console.WriteLine("hit any key to exit...");
Console.ReadKey();
#endif
}
}
}

```

Running this on my Thunderbird email files generates the following CoolingStats.txt file.

```

2022/05/15,05:46,58,0,92,0,0,0,0
2022/05/15,12:00,338,33,96,0,0,0,0
2022/05/15,13:00,22,24,96,0,0,0,0
2022/05/15,14:00,0,0,95,0,0,0,0

```

**** LOTS OF DATA REMOVED FOR CLARITY ****

```

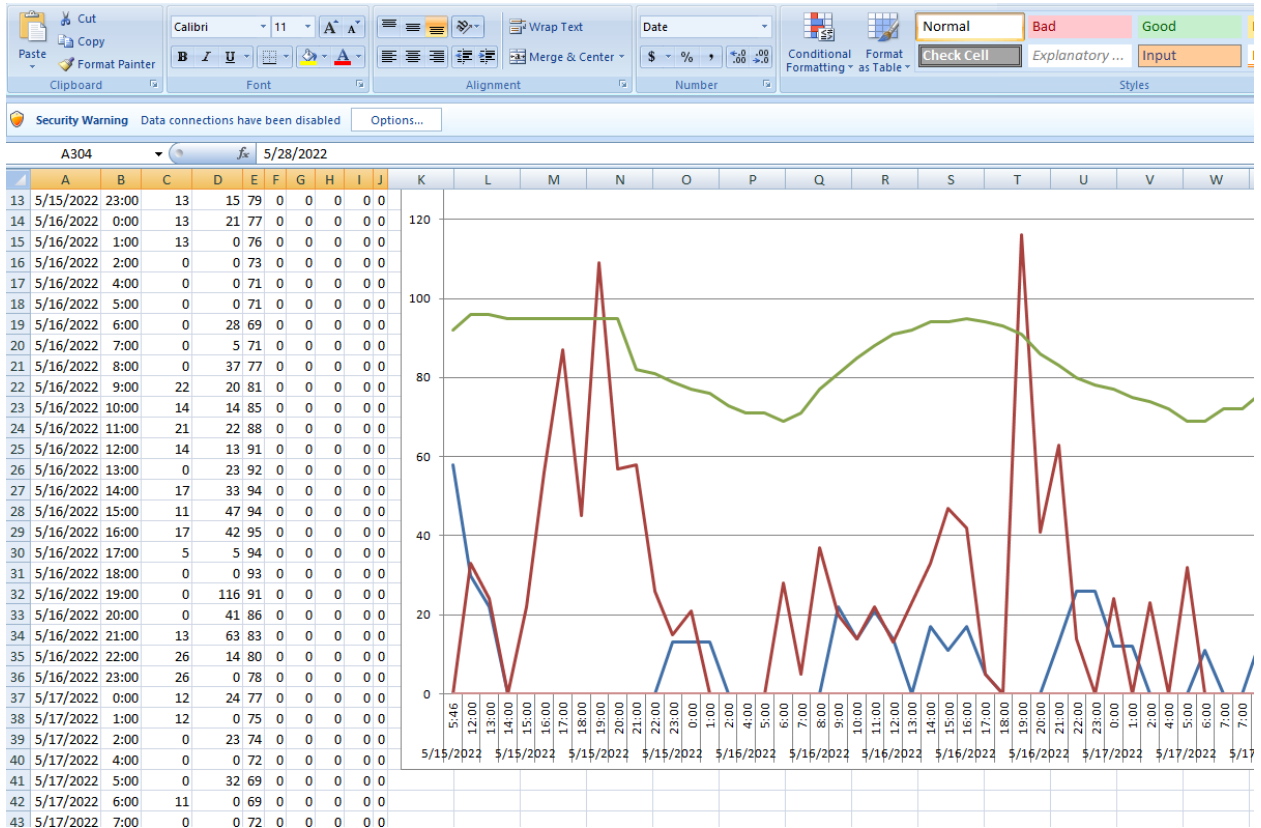
2022/05/29,01:00,12,28,70,0,0,0,0
2022/05/29,02:00,0,5,68,0,0,0,0
2022/05/29,04:00,0,0,66,0,0,0,0
2022/05/29,05:00,0,0,66,0,0,0,0
2022/05/29,06:00,0,0,66,0,100,749,568,0
2022/05/29,07:00,213523,213523,68,5,3,2,3,0
2022/05/29,08:00,25,11,71,0,0,61,0,0

```

Graphing and Using the Email Data

You can then import CoolingStats.txt into Excel and create a graph of the data using well know methods.

- green line is the outdoor temperature
- red line is run time in minutes for the bed room A/C
- blue line is run time in minutes for the main room A/C



This allows you to easily view data point trends collected in Polisy to determine corrective action. In my case the data points indicated that (I have two separate A/C units in my house) one of my A/C units (the bed room A/C) was working much harder than the other. The fix for me was to adjust the main room A/C thermostat down one degree so that it would turn on more often. Now they are sharing the load properly.