**Paper 117-2010**

# This SAS® Program Says to Google, "What's Up Doc?"

Scott Davis, COMSYS, Portage, MI

## Abstract

When you think of the internet, there are few things as ubiquitous as Google. What may not be quite as well known is Google docs which is a web-based Office-like suite for document creation and management. You can create and manage a number of different documents and store them for free on the internet.

There is a tremendous advantage to using a tool like Google docs for the consumer and for the small business owner alike. That advantage, of course, is the cost. Google docs = FREE. Where does SAS come into play? SAS has the tools (with a little finessing) to retrieve data from one of these documents and turn it into information.

This paper explores using the FILENAME URL statement to retrieve data from a Google docs spreadsheet. The data can then be manipulated, turned into a report and published back out to the internet for your colleagues.

## Introduction

The idea for this paper came from a real-life example.  I had some data that I wanted to collect and I wanted to be able to have few restrictions for how the data entry could occur.  Traditionally, my data source was local and access to it was a non-issue.  The problem arose whenever I was using a secure site using a VPN (Virtual Private Network) and could not access my local data without first logging out of the VPN.

The first part of the solution was easy: to make an entry form using Google docs and save the data as a Google spreadsheet (this is the default behavior for a form object).  Next I imported the data into SAS and voila: we have a conference paper!

## Google Docs

As mentioned above, Google docs provides a variety of options for creating and managing documents and storing them on the web.  You can create documents, spreadsheets and presentations to name a few.  When you upload a file to Google docs the system will ask you if you want to convert the file into a Google docs file or leave it in its native format.  The advantage to converting the file is that you can then modify it from any web browser and share with other users.  A disadvantage is that some settings/features that you have in the file may not be able to be converted, e.g. complex data validation in a spreadsheet would probably not translate.

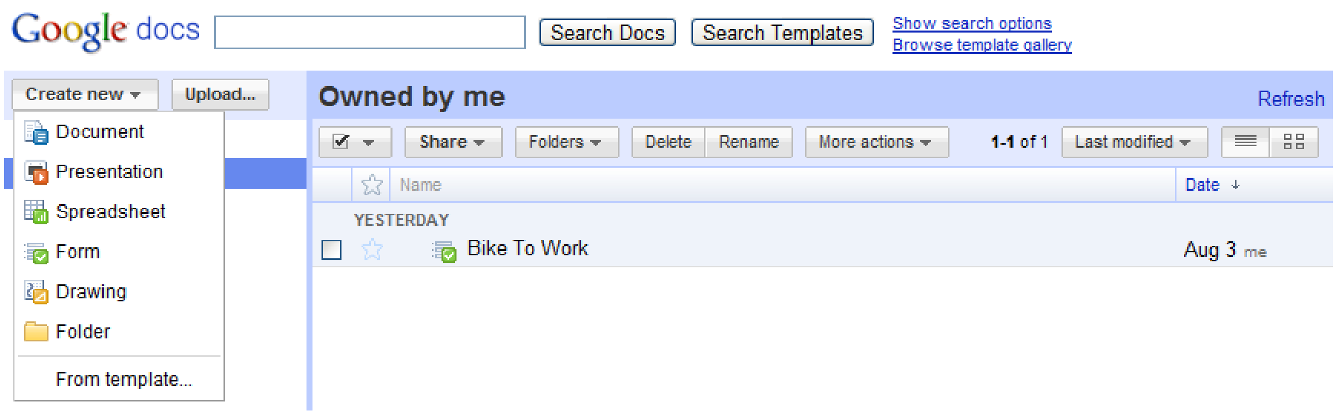


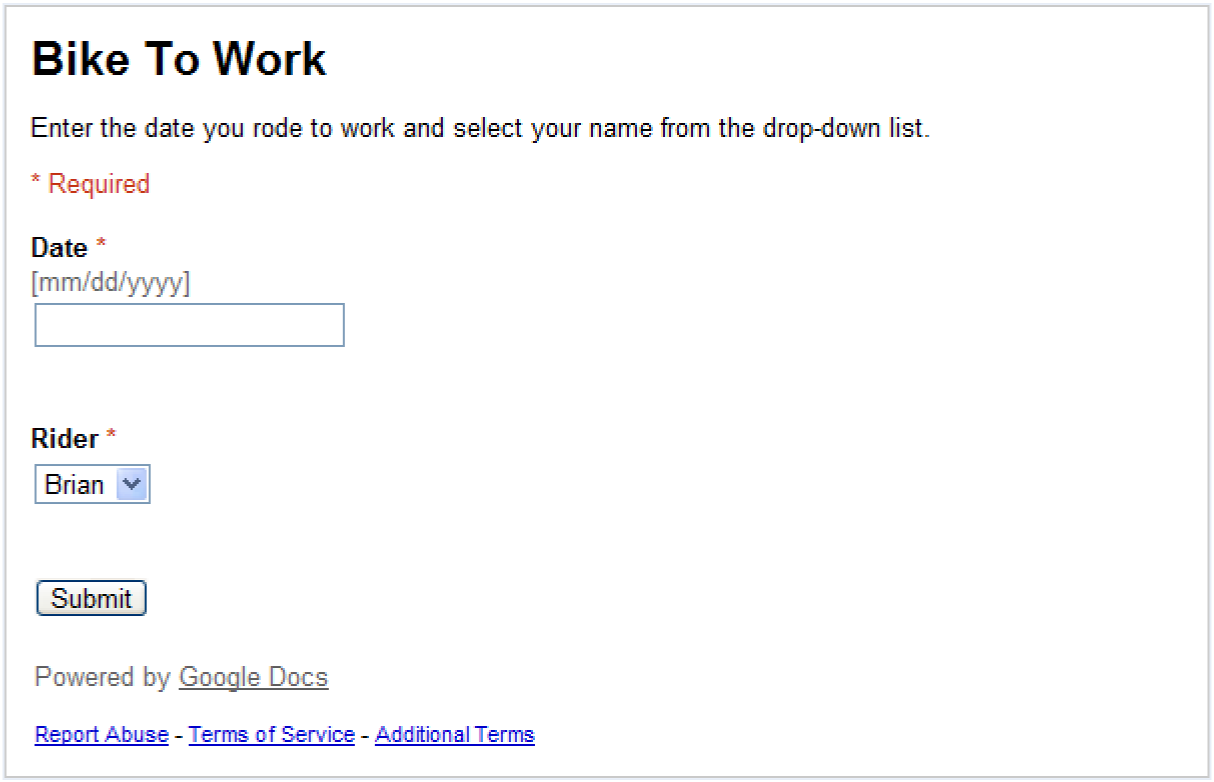
**Figure 1**

A nice feature about using the forms document type is that you don't have to have a spreadsheet behind it created when you design your form.  Once the form design is complete, Google docs automatically creates a corresponding spreadsheet repository behind the scenes.

I found that there are some limitations in using the form/spreadsheet pairing in Google docs.  For example, one of my fields on the form is a date. A date object does not exist so I had to use a text object for this field.  You do have the ability to provide entry instructions to give guidance, but that will only get you so far with getting valid data.  On the spreadsheet side of things you can set data validation for a field to be a valid date.  It's not ideal, but it does give you the opportunity to 'check' your data before bringing it into SAS.  For these reasons, the methods set forth in this paper are definitely not applicable for every situation, especially large scale operations, but there are times where it just might work for you.

The form that I created is a simple one to collect information on who is riding their bike to work.  We have a small group of people in our office that regularly ride our bikes to work and this is one of the ways we've experimented with keeping track of our rides.

Here is the bike rider form:



Bike To Work

Enter the date you rode to work and select your name from the drop-down list.

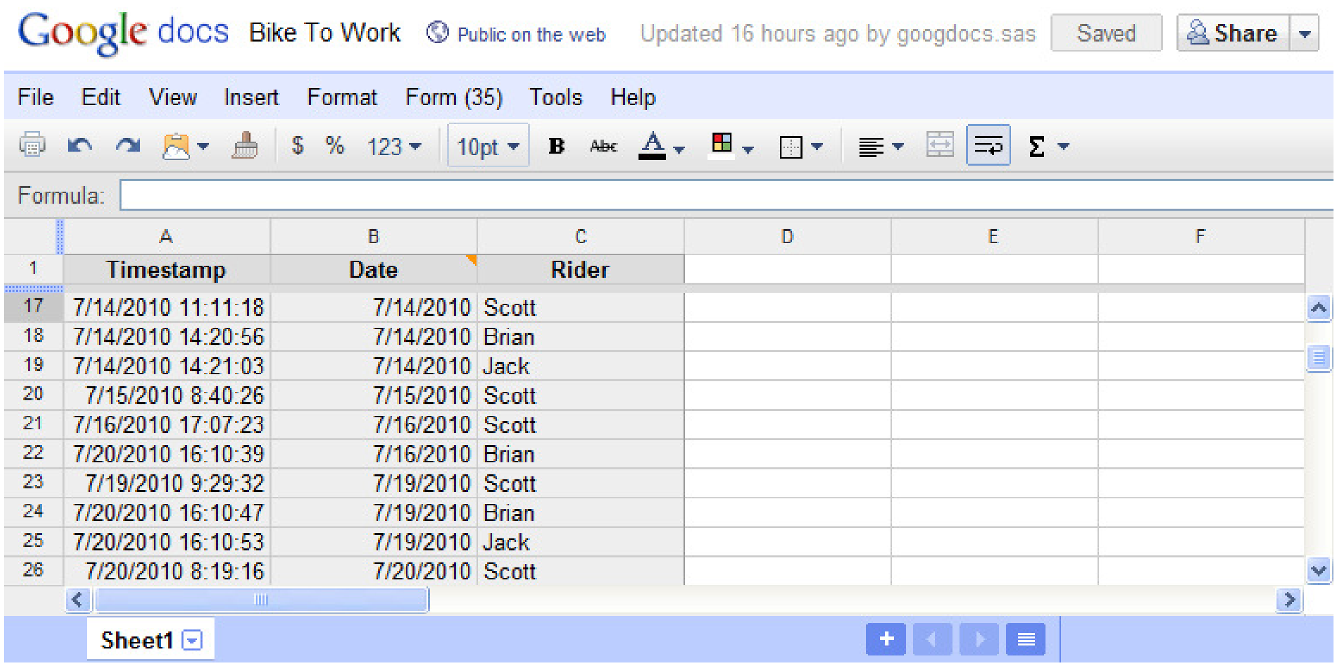
* Required

Date *
[mm/dd/yyyy]

Rider *
Brian

**Figure 2**

The following is a partial view of the spreadsheet data:



| | A | B | C |
|---|---|---|---|
| 1 | Timestamp | Date | Rider |
| 17 | 7/14/2010 11:11:18 | 7/14/2010 | Scott |
| 18 | 7/14/2010 14:20:56 | 7/14/2010 | Brian |
| 19 | 7/14/2010 14:21:03 | 7/14/2010 | Jack |
| 20 | 7/15/2010 8:40:26 | 7/15/2010 | Scott |
| 21 | 7/16/2010 17:07:23 | 7/16/2010 | Scott |
| 22 | 7/20/2010 16:10:39 | 7/16/2010 | Brian |
| 23 | 7/19/2010 9:29:32 | 7/19/2010 | Scott |
| 24 | 7/20/2010 16:10:47 | 7/19/2010 | Brian |
| 25 | 7/20/2010 16:10:53 | 7/19/2010 | Jack |
| 26 | 7/20/2010 8:19:16 | 7/20/2010 | Scott |

**Figure 3**

You'll notice that although the form only has two fields, the spreadsheet has three columns. This is because the timestamp field is automatically generated due to the link between the form and the spreadsheet. In my program I chose to ignore the timestamp field because it was not of any use to me, but it could be helpful in certain situations.

## The Program

The core piece of the program is using the FILENAME statement with the URL option. Not coincidentally, this was also the trickiest part of the program to get to work. Google docs uses a secure protocol so the URL is preceded with http<u>s</u> with the s representing the secure nature of the site. As I was developing this paper I found, by working with SAS Tech Support, that for this particular site the FILENAME URL did not like the 's' part of the protocol, so in order to get it to work, I had to have the URL point to just http. This initially allowed me to gain access to the spreadsheet, but then another problem reared its head.

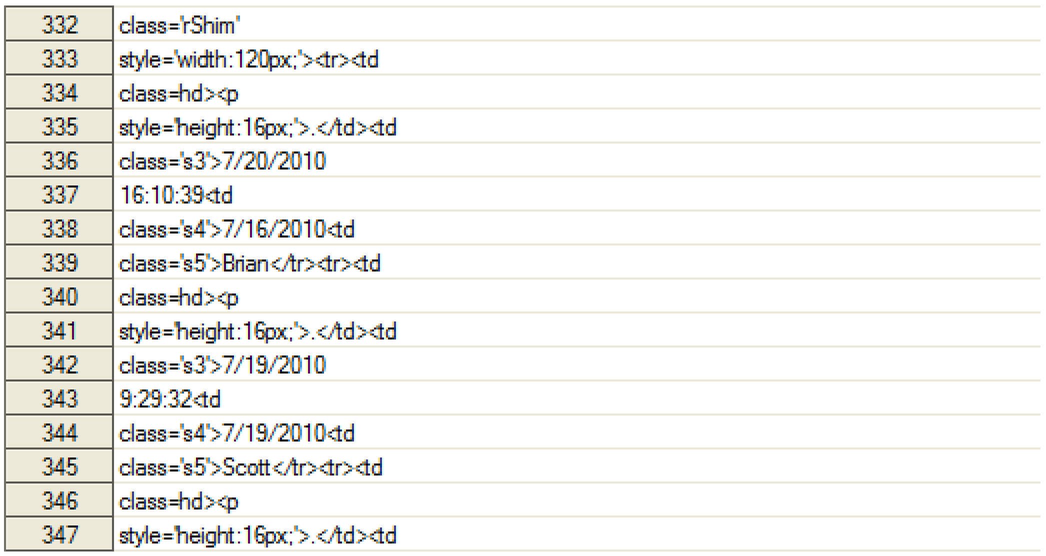
What was discovered next, again in concert with SAS Tech Support, was that I needed to have a proxy server running to be able to read in the html. SAS Tech Support informed me that at the time of this writing, there is a defect with the URL engine where SAS is unable to properly parse the header section for some websites. The use of the proxy server normally masks the defect. A corresponding SASNote has been created referencing this issue and can be found here: http://support.sas.com/kb/40538.

My company did not have a proxy server running so after doing a search I found a program called Fiddler that sets up a debugging proxy server for you. With this running in the background I was then able to pull the html into SAS.

This is what the FILENAME URL looks like:

```
filename foo url
'http://spreadsheets.google.com/ccc?key=0Apl4Rhfuebd9dFV4c2tHbXl6cEF1eWFFbHJPNHVMT1E&h
l=en#gid=0'
proxy='http://127.0.0.1:8888'
debug;
```

The FILENAME URL pulls in all of the html. As you may imagine with a web-based spreadsheet there is a lot of html that we don't need before we get to the actual data that we want to bring into SAS. The next step is to filter through the raw data to find what we want to keep. As I looked at the raw html (see Figure 4) it was readily apparent what the pattern was for the relevant data.



| | |
|---|---|
| 332 | class='rShim' |
| 333 | style='width:120px;'><tr><td |
| 334 | class=hd><p |
| 335 | style='height:16px;'>.</td><td |
| 336 | class='s3'>7/20/2010 |
| 337 | 16:10:39<td |
| 338 | class='s4'>7/16/2010<td |
| 339 | class='s5'>Brian</tr><tr><td |
| 340 | class=hd><p |
| 341 | style='height:16px;'>.</td><td |
| 342 | class='s3'>7/19/2010 |
| 343 | 9:29:32<td |
| 344 | class='s4'>7/19/2010<td |
| 345 | class='s5'>Scott</tr><tr><td |
| 346 | class=hd><p |
| 347 | style='height:16px;'>.</td><td |

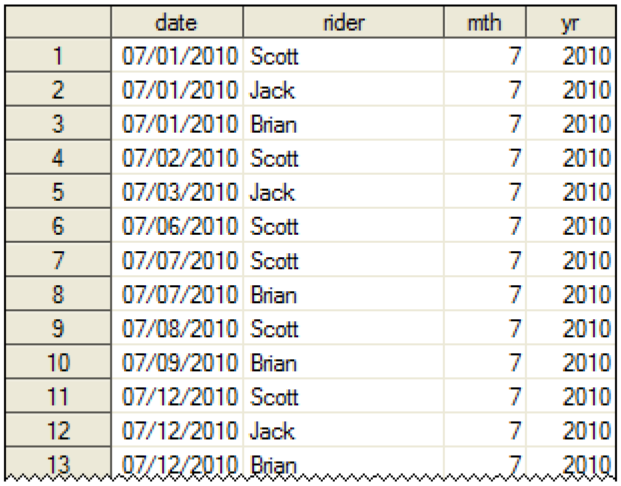
**Figure 4**

Recognizing a pattern helps ensure that the data will be a good candidate for using regular expressions. I needed to create two regular expressions, one for the date variable and one for the name variable.

Here are the two regular expressions I used to parse the html:

```
        regex1=prxparse("/class='s\d+'>(\d+\/\d+\/\d+)<td/");
        regex2=prxparse("/class='s\d+'>([A-Z][a-z]+)/");
```

Once the regular expressions have been applied, the data have been filtered down to just the pieces I need. All of the other html header, body and miscellaneous tags are removed. The only problem left for me now is that my records are split and I want to join every two records together to form one row. Throw in the MOD function and there is my final data set ready for reporting/analysis:

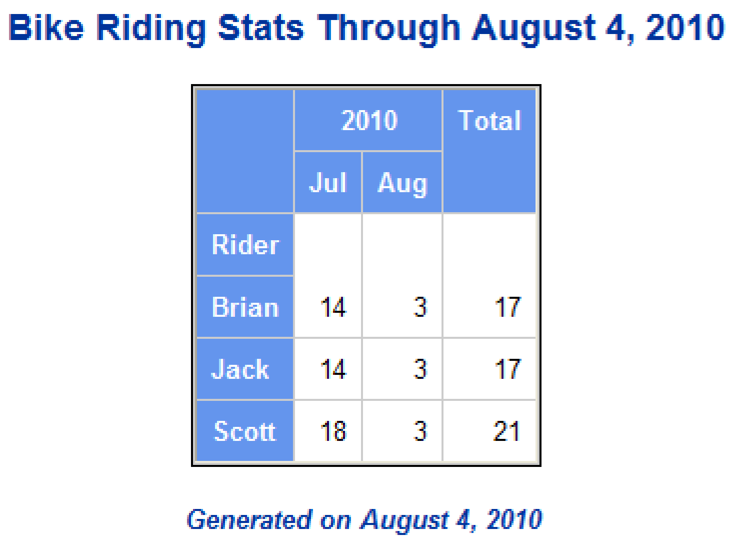


| | date | rider | mth | yr |
|---|---|---|---|---|
| 1 | 07/01/2010 | Scott | 7 | 2010 |
| 2 | 07/01/2010 | Jack | 7 | 2010 |
| 3 | 07/01/2010 | Brian | 7 | 2010 |
| 4 | 07/02/2010 | Scott | 7 | 2010 |
| 5 | 07/03/2010 | Jack | 7 | 2010 |
| 6 | 07/06/2010 | Scott | 7 | 2010 |
| 7 | 07/07/2010 | Scott | 7 | 2010 |
| 8 | 07/07/2010 | Brian | 7 | 2010 |
| 9 | 07/08/2010 | Scott | 7 | 2010 |
| 10 | 07/09/2010 | Brian | 7 | 2010 |
| 11 | 07/12/2010 | Scott | 7 | 2010 |
| 12 | 07/12/2010 | Jack | 7 | 2010 |
| 13 | 07/12/2010 | Brian | 7 | 2010 |

**Table 1**

## Output

Of course with ODS and the various tagsets we have many options for generating the output. To stick with the theme of web-based data collection and reporting I chose to use ODS HTML. I applied a format, did a little tweaking of the date fields and came up with this table to show who is riding when:



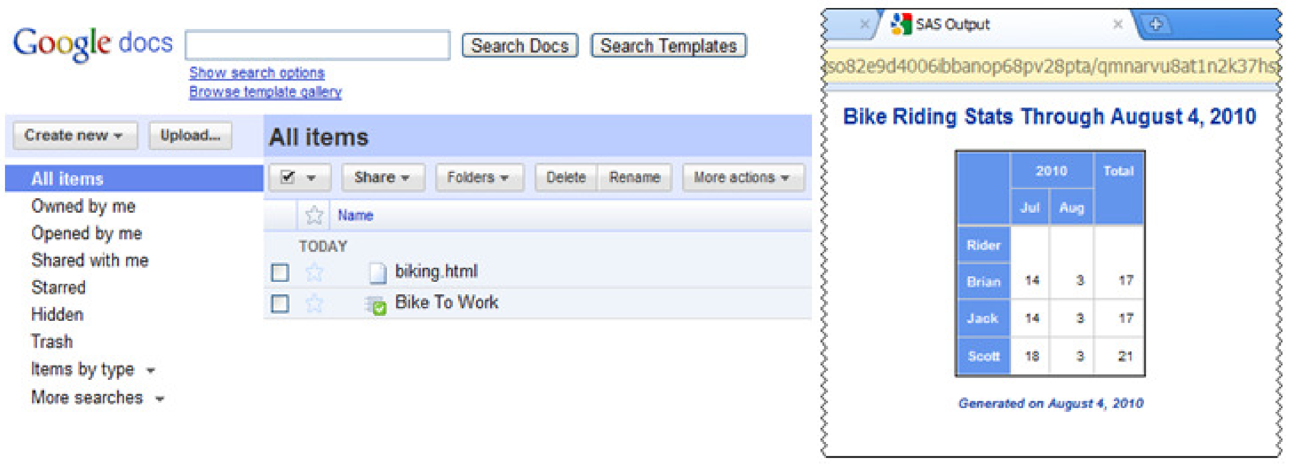
Bike Riding Stats Through August 4, 2010

| | 2010 | | Total |
|---|---|---|---|
| | Jul | Aug | |
| Rider | | | |
| Brian | 14 | 3 | 17 |
| Jack | 14 | 3 | 17 |
| Scott | 18 | 3 | 21 |

Generated on August 4, 2010

**Figure 5**

The next part is not terribly sophisticated though I am sure it could be. In order to bring things full circle with my Google docs, I upload the file and can share it with all the participants (in my case the other riders).



Bike Riding Stats Through August 4, 2010

| | 2010 | | Total |
|---|---|---|---|
| | Jul | Aug | |
| Rider | | | |
| Brian | 14 | 3 | 17 |
| Jack | 14 | 3 | 17 |
| Scott | 18 | 3 | 21 |

Generated on August 4, 2010

**Figure 6**

## Conclusion

This paper demonstrates one way in which an individual or group may easily generate the means for collecting data on the web, pull it into SAS and then publish it back out for users.

There are a variety of factors that would influence one's decision to try this approach. These factors include, but are not limited to, form design, data to be collected and corporate security policies. With the right environment though this method of data collection and reporting using the web without expending extra resources can be very useful.

## Acknowledgements

I would like to thank Rosie Grzebyk and Debbie Vandervest for their support in reviewing this paper.

## Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Scott Davis
COMSYS
5220 Lovers Lane, Suite 200
Portage, MI 49002
Work Phone: 269-553-5122
E-mail: sdavis@comsys.com