Paper 001-2013

Floating on Cloud 9.3: Leveraging the Cloud With SAS and Google Drive

William G. Roehl, MarketShare, Los Angeles, CA

ABSTRACT

Our organization has been utilizing Google® Drive® (previously: Google Docs®) to keep project documentation centrally located for ease of access by any user on any platform. Up until this point, SAS® developers had to manually import and/or export datasets to/from flat files or Microsoft® Excel® in order to update data stored in the cloud.

This paper provides a powerful macro toolkit which facilitates direct access to Google Spreadsheets through its published API allowing uploading, downloading, deletion and live data manipulation of cloud-based data. This provides an opportunity for a significant reduction in the amount of manual work and time required for SAS developers to perform these basic functions.

Code was developed with SAS 9.3 and HTMLTidy (25MAR2009) running under Microsoft Windows® 7.

INTRODUCTION

Google Drive provides a low cost/free mechanism for effective data sharing across an organization. Coupled with many of the standard tools available through any other office product suite, Google has provided a collaborative framework which permits the ease of editing documents and data on any platform by any number of individuals. Utilizing these tools is easy enough for any organization; however enabling SAS to directly interact with documents available on Google Drive's Spreadsheets becomes quite a bit more difficult, especially when automation is key.

One of the biggest functions SAS developers in the organization desired was the ability to directly manipulate Google Drive's Spreadsheet data. This includes uploading and downloading spreadsheets, adding new worksheets, and updating data in existing spreadsheets. While Google Drive's API functionality is wide ranging, due to our organizational requirements, this paper covers only the necessary functions and their related API calls used together to deliver a working final product.

The macros included in the toolkit as part of this paper are:

%SAS_SETHEADER	Set PROC HTTP headers
%GDOC_AUTH	User authentication
%GDOC_GETLIST	Create a list of available documents
%GDOC_GETDATA	Download a document
%GDOC_CREATEDOC	Create a new document
%GDOC_PUTDATA	Replace data in a document
%GDOC_ADDSHEET	Add a worksheet/tab
%GDOC_UPDATECELL	Update an individual cell's contents
%GDOC_ADDROW	Add a row to a spreadsheet
%GDOC_DELETESHEET	Delete a worksheet
%GDOC_CREATEWS	Populate a preexisting worksheet with a dataset

Many of the macros included in this toolkit require Ted Clay's %do_over and %array macros to operate. These help with repetitive processing and both %do_over and %array increase readability and decrease overall code line length. The paper which presents these two macros is included in the References section along with a link to the macros themselves and will need to be available to the developer in order for this toolkit to run properly. In addition to these macros, Dirk Paehl's HTMLTidy will need to be made available to help process the XML into a readable format by the DATA step.

UTILIZING GOOGLE® DRIVE SPREADSHEETS FOR DATA SHARING IN THE CLOUD

Let's assume a project document is already available on Google Drive as a spreadsheet. In our organization this document will have numerous tabs serving a variety of purposes, one of which is offering a list of variables and their meaning so other teams can easily access those data. The organization's workflow is an iterative process and this document may be updated several times a day. The prior methodology was to process the data dictionary in SAS and either open it in Excel and copy and paste into Google Drive or save out to an Excel document and upload it via Google Drive's web interface. While this does not seem like a complex process, it does take time and due to potential changes on the Google Drive end from consumers of these data, it made sense to create a system in which a potentially updated version of the document could be imported directly from Google Drive into SAS, checked for changes, modified, and placed back into the cloud with minimal manual processing by SAS developers.

In order to complete this process with the API framework and toolkit as it is currently developed, a developer would: authenticate, get the current document list and determine if the worksheet already exists, either replace the worksheet with new data or create a new worksheet and populate it with a SAS dataset. While there are other options available to a developer as part of this toolkit, this paper will provide more in-depth explanations of this single working example.

%GDOC_AUTH

Parameter	Description
U	Google Accounts username (e-mail address: GMail or work)
P	Google Accounts password

Example:

```
/* Authenticate */
%gdoc_auth(username@gmail.com,password)
```

The %GDOC_AUTH macro builds two different URLs from the arguments passed (username, password) and utilizes PROC HTTP to execute the API call. Once the PROC HTTP header file is created, URLs are passed, and the username and password are accepted, Google returns two authentication values which are then stored in global macro variables &AUTH (general Google Drive authentication) and &AUTH_S (Google Drive Spreadsheets authentication) for later use in the program.

%GDOC_GETLIST

Parameter	Description
N/A	This macro takes no arguments.

Example:

```
%gdoc_getList()
```

This macro calls out to the Google Drive API and returns a list of 'files' available to the authenticated user as well as a list of available spreadsheets. It processes the XML response received from the API and creates two datasets of important URLs necessary to later interact with these files. The XML processing is done with regular expressions such as in the example excerpt below:

```
/* Grab URLs for the documents and strip extraneous tags */
data gdoc_list_new(where=(id ^= ''));
      […]

      /* Edit link */
      if prxmatch ('/resumable-create-media/',line) then do;
            line = prxchange("s/^.*href=.?//",-1,line);
            line = prxchange("s/'.\/>.*$//",-1,line);
            id = 'create_link';
      end;

      […]
run;
```

After the XML data are processed, the resulting datasets are transposed so that each 'file' record appears on one row in the final dataset. While SAS' XMLMapper tool can be utilized instead of using regular expressions to parse the XML result sets, by utilizing in-line code the number of additional files required during deployment are less numerous and possibly less nebulous.

Example document list datasets created (subset):

Figure 1

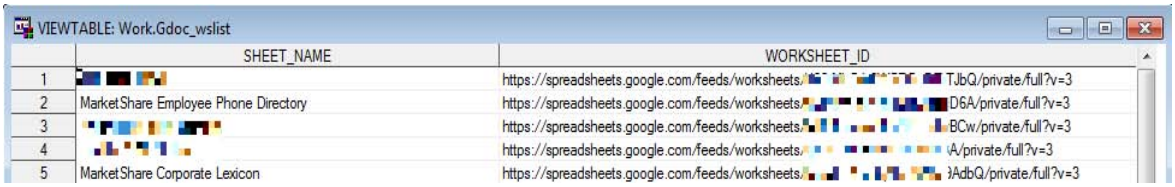

Figure 2

%GDOC_CREATEDOC

Parameter	Description
TITLE	Title of the new document
DATASET	SAS dataset to populate new document with (Sheet1)

Example:

```
/* Create a new spreadsheet and upload dataset data */
%gdoc_createDoc(title=%str(SAS Global Forum Test), dataset=SASHELP.DFTDICT);
```

This macro call will first create a new blank Google Drive Spreadsheet named “SAS Global Forum Test” and populate it with the contents of SASHELP.DFTDICT. The macro creates an XML payload file which contains the information required by the API to create a document and sends it off to the API to be processed. If the payload insertion was successful, the API returns more XML which provides a URL which will allow further processing on the newly created spreadsheet. It’s at this time the passed dataset is converted to TSV (Tab Separated Values) and uploaded onto the blank spreadsheet via the API.

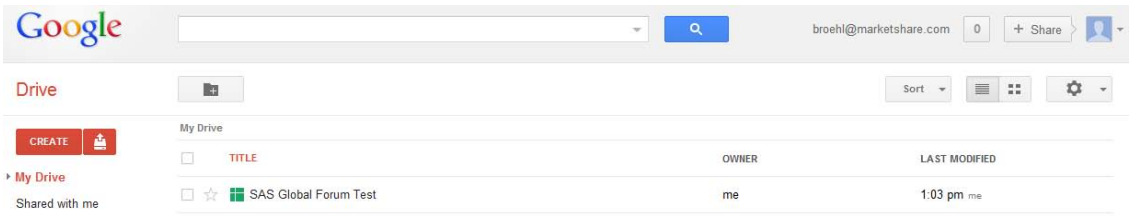

Figure 3

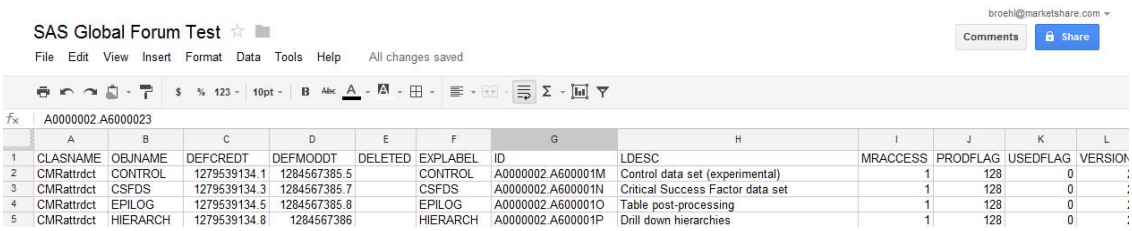

Figure 4

ADDING A WORKSHEET TO A PREEXISTING SPREADSHEET

Some of the other available functions, namely %gdoc_addSheet, require a little more legwork and runtime to make a useable document due to limitations present in the API. The process for adding a sheet includes pulling the column names for the dataset from the dictionary tables (SASHELP.VCOLUMN), changing them to lowercase, inserting the column names cell by cell, and then uploading the remainder of the dataset row by row. While this is time consuming

due to the number of calls which must be made, it still allows for a level of automation which is not available when manual uploads must be done.

Utilizing a combination of several of the developed macros available in this toolkit, a final %gdoc_createWS macro was developed as a wrapper to deliver an easy way for end-users to create a new worksheet in an existing spreadsheet. This macro uploads the contents of a dataset to the new sheet and using a single macro call, all of the complex work required to create a new worksheet and upload it to Google Drive is completed for the developer.

The macro first reads all of the data columns from the SAS dictionary tables and makes them lowercase as is required by the API. It then creates a new blank worksheet in a preexisting spreadsheet by specifying the calculated number of rows and columns and finally adds the column names to the new worksheet column by column.

After that is complete, the macro reads each dataset line by line and creates an individual PAYLOAD.XML file to upload one at a time into the newly created worksheet. When it is done, the macro cleans up the temporary PAYLOAD*.XML files and you're left with a workable spreadsheet in the cloud.

Example:

```
/* Create a new spreadsheet and upload dataset data */
%gdoc_createWS(SAS Global Forum Test, SASHELP_CARS_SHEET, CARS, SASHELP);
```

The result of this operation is a new blank worksheet (tab) named "SASHELP_CARS_SHEET" is created and the contents of SASHELP.CARS is uploaded under "SASHELP_CARS_SHEET" in the "SAS Global Forum Test" spreadsheet. In this particular instance there are 443 calls made to the API; 15 for the individual column headers and 428 rows for a total of 429 rows in the spreadsheet. Timing the entire manual process took, on average, around 10 minutes to export the dataset to Excel, create the sheet in Google Drive and then upload these data. The same process using the macro suite outlined in this paper took a little under three minutes on average. While 7 minutes may not seem like a large time savings for one single run, used as a repetitive process sometimes running several times per day, this can create real man-hour savings and one which is definitely worth exercising.

REFERENCES

- Ted Clay. (2006). Tight Looping With Macro Arrays. http://www2.sas.com/proceedings/sugi31/040-31.pdf http://www.sascommunity.org/mwiki/images/e/ec/Clay-TightLooping-macros.zip
- Google Inc. (2012). Google Drive List/Spreadsheets APIs v3.0. https://developers.google.com/google-apps/
- Dirk Paehl. (2009). HTML Tidy for Win32. http://www.paehl.com/open_source/?HTML_Tidy_for_Windows

CONTACT INFORMATION

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

William Roehl
MarketShare
11150 Santa Monica Blvd
Los Angeles, CA 90025
broehl@marketshare.com / @garciasn / http://www.linkedin.com/in/billroehl

APPENDIX A.

Code repository: https://github.com/billroehl/google_drive_api_sas