Paper 049-2010

Bridging the Gap between the Google Analytics API and SAS®

William G. Roehl, Capella University, Minneapolis, MN

ABSTRACT

Third parties which collect your data generally offer an acceptable reporting piece which provides the general public with a good sense of the how, when, where, and why. For those of us that want to dig a little deeper, several offer a fairly easy way to access this data and feed it into different programs to do our own analysis.

This paper will illustrate a drop-in method, utilizing a few simple macros and cURL that may be utilized to import website analytics data from Google Analytics directly into SAS® data sets for further analysis. The intended audience for this paper is SAS® developers with medium to advanced knowledge of SAS® and its XML Mapper software package.

Code was developed with SAS® 9.2 and cURL 7.19.6-SSL running under Windows XP Professional.

INTRODUCTION

Google Analytics is a free online statistics monitoring service which provides very detailed information about a website. It offers a way for website authors and marketers to collect an inordinate amount of data about their website, their users, and the advertising they do and then manipulate this data it in various ways which are predetermined by those that have developed the reporting piece. While the default reporting tools available are quick and easy to understand, they do not allow you to get as deep down inside the data as some may prefer.

While there is extensive documentation provided by Google about their Analytics API, it is sometimes easier if all the heavy lifting is done by SAS® and the only items the user has to provide are the details. Using two easy macros, interfacing with the Google Analytics API becomes a cinch and aggregating the data becomes a matter of a few SAS® procedures and DATA steps!

The work is separated out into two macros. One to login and retrieve the authentication key from Google via the HTTP procedure in SAS® 9.2 and the second which talks with the Google Analytics API to send and receive information via cURL at the command line.

Some may question the decision to use cURL instead of SAS® 9.2's built in PROC HTTP to do the bulk of the work bridging the gap between SAS® and the Google Analytics API. The reason cURL was chosen to do the majority of the work in communicating with the API was because during testing I was unable to pass the Google authentication code via the header-in portion of PROC HTTP. This inability caused each subsequent API call to fail, not returning any results and instead only caused HTTP error codes from the web server to be displayed in SAS®. After moving to cURL and using the same settings, the communication worked and the data was easily passed between SAS® and the Google Analytics API.

%GA_AUTH MACRO PARAMETERS

Parameter	Description	Default
U	Google username (e-mail address)	N/A
P	Google password (sent over SSL but shown plaintext in code)	N/A

Example: `%GA_AUTH(username@gmail.com,password);`

Google provides a variety of different authentication methods in which to pass the user's credentials to the API but the one most appropriate for this use (according to the API documentation: a single-user application running on a computer and not the web) is ClientLogin. In order to successfully employ this authentication method, you simply need to provide both the user's e-mail address and password to which Google returns an authentication token back to the program. To accomplish this, the %GA_AUTH macro passes the user's login details through to the API and then parses the authentication token out of the returned string and stores it as a global macro variable for subsequent use throughout the rest of the program.

%GA_GETXML MACRO PARAMETERS

Parameter	Description	Default
XML_PATH	Location of the response XML file output by cURL	N/A
XML_MAP	Location of the XML Map file created by XML Mapper	N/A
GA_FEED_TYPE	Specifies the type of API call to make (ACCOUNT, FEED)	ACCOUNT
START_DT	Start date of data query	2009-01-01
END_DT	End date of data query	2009-01-31
DIMENSION_LST	List of Google Analytics dimensions to query	ga:date
METRIC_LST	List of Google Analytics metrics to query	ga:visits
SORT_LST	List of Google Analytics metrics or dimensions to sort ON	-ga:date (DESC)
PROFILE_ID	Google Analytics profileID (RETURNED BY ACCOUNT query)	N/A
PRETTYPRINT_BOOL	Boolean value for indention of XML elements	TRUE

Example: `%GA_GETXML (c:\temp\response.xml, c:\temp\maps\GA_FEED.map, ACCOUNT);`

The %GA_GETXML macro is the second macro which the program uses to communicate with the Google Analytics API. While it is a short one, the macro relies heavily on the parameters that the user chooses to pass to it. In the example above, the macro call retrieves the account feed information which provides a handful of user-specific observations including the account name, ID, and the profileID. The profileID is the most important piece which we need to pass through to all future API calls when requesting specific metrics.

After retrieving the account feed, the macro parses the resulting XML through a previously created XML map which was created utilizing XML Mapper. The macro then places the resulting data into a SAS® data set of the same name as the XML map. A simple DATA step and call symputx statement creates a PROFILE_ID macro variable which is then passed along with all subsequent calls for data.

Example:
```
%GA_GETXML  (c:\temp\response.xml
            , c:\temp\maps\GA_METRICS.map
            , DATA
            , start_dt=2009-08-01
            , end_dt=2009-08-31
            , dimension_lst=%str()
            , metric_lst=%str(ga:visits,
                              ga:pageviews,
                              ga:entrances)
            , sort_lst=ga:visits
            , profile_id=&GA_PROFILEID.
            , prettyprint_bool=true);
```

The %GA_GETXML macro, in addition to retrieving the PROFILE_ID from the account feed also grabs specific metrics and dimensions from the Google Analytics API which the user is then able to manipulate in many ways using SAS®' various data reporting procedures. The Google Analytics API documentation goes into great detail and lists all possible options which may be requested, via the above macro, to return the desired data. In the examples that follow, only a few of the many possible combinations are explored.

USING THE GOOGLE ANALYTICS MACROS IN PRACTICE

After running the code you are left with raw data straight from Google Analytics which is easily reported on using any of the SAS® tools you have at your fingertips (e.g. the SAS® GPLOT procedure). The following code snippets show two quick examples of what you can do with the data returned by Google Analytics' API.

DATA STEP

The first example of how to report the data returned from the Google Analytics API would be a simple DATA step. Let's take a look at what is stored in a data set after the API is asked to return a variety of metrics about general site usage:

ga_visits	ga_pageviews	ga_entrances	ga_bounces	ga_timeOnSite	ga_exits	ga_newVisits
20871	35697	20863	14018	2613984	20863	13518

Aside from ga_timeOnSite which is returned in seconds, the data returned for each of the metrics are straight counts of visitors and how they interacted with the website being profiled. Using some simple formulas provided in the Google Analytics API documentation we are able to give a brief overview of the traffic received on the site during the given time frame.

```
data report(drop=ga:);
    set ga_metrics_xpose;
    visits = ga_visits; pageviews = ga_pageviews; new_visits = ga_newVisits;

    pagesPerVisit = put(pageviews/visits,4.3);
    bounceRate = put(ga_bounces/ga_entrances, percent8.2) ;
    avgTimeSite = put(ga_timeOnSite/ga_visits,TOD.);
    pctNewVisits = put(new_visits/visits,percent8.2);
run;
```

Output:

visits	pageviews	new_visits	pagesPerVisit	bounceRate	avgTimeSite	pctNewVisits
20871	35697	13518	1.71	67.19%	00:02:05	64.77%

PROC GPLOT

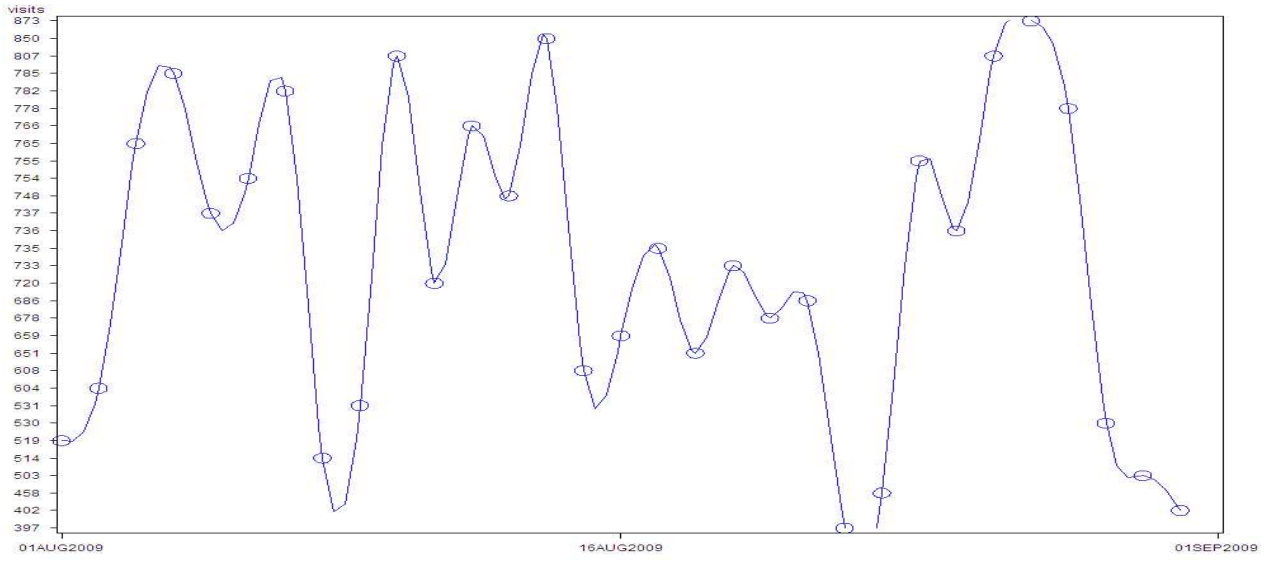
The second easiest to understand example for showing how to utilize the data returned by the API would be to employ PROC GPLOT to display a graph of the data. After asking the API to provide the number of visits, per day, for the range of August 1st, 2009 through August 31st, 2009 we are presented with a simple table. Please note this is only a subset of the monthly data and the SAS_DATE field is calculated in a DATA step and not returned by the API itself:

NAME	DATE	VISITS	SAS_DATE
ga:visits	20090801	519	01AUG2009
ga:visits	20090802	604	02AUG2009
ga:visits	20090803	765	03AUG2009
ga:visits	20090804	785	04AUG2009

```
proc gplot data=ga_visitsbyday;
    symbol i=spline v=circle h=2;
    plot visits*sas_date;
run;
quit;
```

The graph below plots the visits by date to give a nice visual representation of the trends exhibited by the data. The x-axis covers the time period of the graph and the y-axis is the number of visits. Using this information you can view the cyclical nature of the visits with the most number of visits occurring during weekdays and the least during weekends.

Output:



REFERENCES

- Daniel Stanberg. (2009). cURL Manual. http://curl.haxx.se/docs/manual.html
- Google Inc. (2009). Google Analytics Developer Docs. http://code.google.com/apis/analytics/docs/

CONTACT INFORMATION

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

William Roehl
Capella University
225 S 6th St, 9th Fl
Minneapolis, MN 55402
william.roehl@capella.edu
http://www.capella.edu

SAS/GOOGLE ANALYTICS API CODE

```
/* Username and Password */
%macro ga_userpass (u,p);
       %global user password;
       %let user=&U;
       %put &USER;
       %let password=&P;
       %put &PASSWORD;
%mend;

/* Authenticate */
%macro ga_auth (u,p);
              %let user=&U;
              %let password=&P;

              %global auth;
              %global authurl;

              filename request 'c:\temp\analytics\ReqResp\request.txt';
              data _null_;
                infile 'C:\temp\analytics\ReqResp\analytics_auth.txt' lrecl=1000;
                input;
                line=resolve(_infile_);
                file request;
                put line;
                call symputx('AUTHURL',line);
              run;

              /* Due to SSL certs issue use cURL instead */
              %let CURL = curl -k -o c:\temp\analytics\ReqResp\response.txt;
              %let CURLURL = "https://www.google.com/accounts/ClientLogin?&AUTHURL.";
              %let CURLFINAL = &CURL. &CURLURL.;

              options noxwait;
              %sysexec (&CURLFINAL);

              filename response 'c:\temp\analytics\ReqResp\response.txt';
              data _null_;
                infile response lrecl=500 dlm='=';
                format var $8. value $256.;
                input var value;
                if var = 'Auth';
                put value=;
                call symputx('AUTH',value);
              run;
%mend;
```

```
/* Pull XML from Google API via cURL and bring it in as a SAS® data set */
%macro ga_getXML (
       xml_path,
       xml_map,
       ga_feed_type,
       start_dt=2009-01-01,
       end_dt=2009-01-31,
       dimension_lst=ga:date,
       metric_lst=ga:visits,
       sort_lst=-ga:date,
       profile_id=ga:0000000,
       prettyprint_bool=true);

       /* Setup the macro variables from the passed parameters */
       %if "&XML_PATH" eq " " %then %let XML_PATH = C:\TEMP\RESPONSE.XML;
       %if "&XML_MAP" eq " " %then %let XML_MAP = C:\TEMP\MAPS\GA_FEED.MAP;
       %if "&GA_FEED_TYPE" eq " " %then %let GA_FEED_TYPE = ACCOUNT;
       %if "&START_DT" eq " " %then %let START_DT = 2009-01-01;
       %if "&END_DT" eq " " %then %let END_DT = 2009-01-31;
       %if "&DIMENSION_LST" eq " " %then %let DIMENSION_LST = ga:date;
       %if "&METRIC_LST" eq " " %then %let METRIC_LST = ga:visits;
       %if "&SORT_LST" eq " " %then %let SORT_LST = -ga:date;
       %if "&PROFILE_ID" eq " " %then %let PROFILE_ID = ga:0000000;
       %if "&PRETTYPRINT_BOOL" eq " " %then %let PRETTYPRINT_BOOL = true;

       /* AUTH is a global macro variable passed from %ga_login() */
       %let CURL = curl -o &XML_PATH. -k -s --header "Authorization: GoogleLogin
       Auth=&AUTH.";

       /* There are two types of URLs to pass to GA, one for account data, the other
       for regular data */
       %if %upcase(&GA_FEED_TYPE) = ACCOUNT %then %let URL = --url
       "https://www.google.com/analytics/feeds/accounts/default?prettyprint=&PRETTYPRIN
       T_BOOL.";

       %if %upcase(&GA_FEED_TYPE) = DATA    %then %let URL = --url
       "https://www.google.com/analytics/feeds/data?start-date=&START_DT.&end-
       date=&END_DT.&dimensions=&DIMENSION_LST.&metrics=&METRIC_LST.&sort=&SORT_LST.&id
       s=ga:&PROFILE_ID.&prettyprint=&PRETTYPRINT_BOOL.";

       /* Build the command line to pass to cURL and execute */
       %let command = &CURL. &URL.;

       options noxwait;
       %sysexec (&command.);

       /* Import the XML cURL retrieved from Google Analytics into a SAS data set */
       /* Please note: the SAS® XML Map and data set name must match the &XML_MAP
       passed (e.g. GA_VISITSBYDAY) */
       filename  response "&XML_PATH.";
       filename  SXLEMAP "&XML_MAP.";
       libname   response xml xmlmap=SXLEMAP access=READONLY;

       data _null_;
              call symputx ('dataset',scan(scan("&XML_MAP",-1,"\"),1,"."));
       run;
       DATA &dataset.; SET response.&dataset.; run;
%mend;
```

```
/*****************************************************************************\
Setup, login, and get auth key from Google
\*****************************************************************************/;
%ga_auth(username@gmail.com,password);

/*****************************************************************************\
Get the account feed information and then make the table useful
\*****************************************************************************/;
%ga_getXML (c:\temp\response.xml, c:\temp\maps\GA_FEED.map, ACCOUNT);

` transpose data=ga_feed out=ga_feed_xpose(drop=_:);
      id name;
      var value;
run;

/*****************************************************************************\
Get Google Analytics overall stats and import them into SAS®
\*****************************************************************************/;
data _null_;
      set ga_feed_xpose;
      call symputx('GA_PROFILEID',ga_profileId);
run;

      %ga_getXML (c:\temp\response.xml,
                  c:\temp\maps\GA_METRICS.map,
                  DATA,
                  start_dt=2009-08-01,
                  end_dt=2009-08-31,
                  dimension_lst=%str(),
                  metric_lst=%str(ga:visits,
                                  ga:pageviews,
                                  ga:entrances,
                                  ga:bounces,
                                  ga:timeOnSite,
                                  ga:exits,
                                  ga:newVisits),
                  sort_lst=ga:visits,
                  profile_id=&GA_PROFILEID.,
                  prettyprint_bool=true);

proc transpose data=ga_metrics out=ga_metrics_xpose(drop=_:);
      id name;
      var value;
run;
```

```
/*****************************************************************************\
Get Google Analytics visits by day stats and import them into SAS®
\*****************************************************************************/;
%ga_getXML (c:\temp\response.xml,
            c:\temp\maps\GA_VISITSBYDAY.map,
            DATA,
            start_dt=2009-08-01,
            end_dt=2009-08-31,
            dimension_lst=ga:date,
            metric_lst=ga:visits,
            sort_lst=ga:date,
            profile_id=&GA_PROFILEID.,
            prettyprint_bool=true);

DATA GA_VISITSBYDAY;
      SET GA_VISITSBYDAY;
      format sas_date date9.;

      /* Convert the YYYYMMDD date field to a SAS® date9. */
      sas_date = input(put(date, 8.), yymmdd8.);
run;

/*****************************************************************************\
Plot the chart of visits/day
\*****************************************************************************/;
proc gplot data=ga_visitsbyday;
      symbol i=spline v=circle h=2;
      plot visits*sas_date;
run;
quit;

/*****************************************************************************\
Calculate basic Site Usage metrics (these were provided in the Google Analytics API
docs and appear on the main dashboard)
\*****************************************************************************/;
data report(drop=ga:);
      set ga_metrics_xpose;
      visits = ga_visits; pageviews = ga_pageviews; new_visits = ga_newVisits;

      pagesPerVisit = put(pageviews/visits,4.3);
      bounceRate = put(ga_bounces/ga_entrances, percent8.2) ;
      avgTimeSite = put(ga_timeOnSite/ga_visits,TOD.);
      pctNewVisits = put(new_visits/visits,percent8.2);
run;
```

RESPONSE.TXT

```
accountType=GOOGLE&Email=username@gmail.com&Passwd=password&service=analytics&source
=SAStesting-1.0
```

GA_FEED.MAP

```
<?xml version="1.0" encoding="windows-1252"?>

<!-- ############################################################ -->
<!-- 2009-09-01T11:19:01 -->
<!-- SAS XML Libname Engine Map -->
<!-- Generated by XML Mapper, 9.2.0.000000_v920c_20080125_21893 -->
<!-- ############################################################ -->
<!-- ###  Validation report                                   ### -->
<!-- ############################################################ -->
<!-- XMLMap validation completed successfully. -->
<!-- ############################################################ -->
<SXLEMAP name="GA_FEED" version="1.2">

    <!-- ############################################################ -->
    <TABLE name="GA_FEED">
        <TABLE-DESCRIPTION>dxp:property</TABLE-DESCRIPTION>
        <TABLE-PATH syntax="XPath">/feed/entry/dxp:property</TABLE-PATH>

        <COLUMN name="name">
            <PATH syntax="XPath">/feed/entry/dxp:property/@name</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>32</LENGTH>
        </COLUMN>

        <COLUMN name="value">
            <PATH syntax="XPath">/feed/entry/dxp:property/@value</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>32</LENGTH>
        </COLUMN>

    </TABLE>

</SXLEMAP>
```

GA_METRICS.MAP

```
<?xml version="1.0" encoding="windows-1252"?>

<!-- ############################################################ -->
<!-- 2009-09-01T13:17:33 -->
<!-- SAS XML Libname Engine Map -->
<!-- Generated by XML Mapper, 9.2.0.000000_v920c_20080125_21893 -->
<!-- ############################################################ -->
<!-- ###  Validation report                                   ### -->
<!-- ############################################################ -->
<!-- XMLMap validation completed successfully. -->
<!-- ############################################################ -->
<SXLEMAP name="GA_MAP" version="1.2">

    <!-- ############################################################ -->
    <TABLE name="GA_METRICS">
        <TABLE-DESCRIPTION>dxp:metric</TABLE-DESCRIPTION>
        <TABLE-PATH syntax="XPath">/feed/entry/dxp:metric</TABLE-PATH>

        <COLUMN name="name">
            <PATH syntax="XPath">/feed/entry/dxp:metric/@name</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>32</LENGTH>
        </COLUMN>

        <COLUMN name="type">
            <PATH syntax="XPath">/feed/entry/dxp:metric/@type</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>32</LENGTH>
        </COLUMN>

        <COLUMN name="value">
            <PATH syntax="XPath">/feed/entry/dxp:metric/@value</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>32</LENGTH>
        </COLUMN>

    </TABLE>

</SXLEMAP>
```

GA_VISITSBYDAY.MAP

```
<?xml version="1.0" encoding="windows-1252"?>

<!-- ############################################################ -->
<!-- 2009-09-02T10:58:15 -->
<!-- SAS XML Libname Engine Map -->
<!-- Generated by XML Mapper, 9.2.0.000000_v920c_20080125_21893 -->
<!-- ############################################################ -->
<!-- ###  Validation report                                   ### -->
<!-- ############################################################ -->
<!-- Column (date) in table (GA_VISITSBYDAY) has an XPath outside the scope of the
table path. The contents of this column may not correspond to other row values
and/or may be missing entirely. -->
<!-- XMLMap validation completed successfully. -->
<!-- ############################################################ -->
<SXLEMAP name="GA_VISITSBYDAY" version="1.2">

    <!-- ################################################################ -->
    <TABLE name="GA_VISITSBYDAY">
        <TABLE-DESCRIPTION>dxp:metric</TABLE-DESCRIPTION>
        <TABLE-PATH syntax="XPath">/feed/entry/dxp:metric</TABLE-PATH>

        <COLUMN name="name">
            <PATH syntax="XPath">/feed/entry/dxp:metric/@name</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>32</LENGTH>
        </COLUMN>

        <COLUMN name="date" retain="YES">
            <PATH syntax="XPath">/feed/entry/dxp:dimension/@value</PATH>
            <TYPE>numeric</TYPE>
            <DATATYPE>integer</DATATYPE>
        </COLUMN>

        <COLUMN name="visits">
            <PATH syntax="XPath">/feed/entry/dxp:metric/@value</PATH>
            <TYPE>character</TYPE>
            <DATATYPE>string</DATATYPE>
            <LENGTH>32</LENGTH>
        </COLUMN>

    </TABLE>

</SXLEMAP>
```