The Call Execute Routine with Metadata: A Powerful Tool for Data Loading

R. Scott Alcorn, Early Warning Services LLC, Austin, TX

Abstract

CALL EXECUTE is a data step routine which interacts with the SAS Macro Facility. It is a versatile tool which enables the user to stack numerous SAS statements and run them together. This is an especially useful function in the realm of extracting, transforming, and loading (ETL) numerous SAS datasets in a single folder, without having to manually write repeated macro calls. Because the CALL EXECUTE function interacts with the SAS Macro Facility during the DATA step, the user can pass in metadata directly, in the form of a master list. For our purposes, we will use a character expression to resolve a macro text expression. Using this feature, along with the concept of metadata, makes ETL processing a breeze. This paper will teach a simple method of passing metadata through the CALL EXECUTE routine by example. This example is in no way meant to represent all possible uses of the CALL EXECUTE routine.

Syntax

CALL EXECUTE (*argument*)

Argument can be one of the following:

- A character string, enclosed in quotation marks. *Argument* within single quotation marks resolves during program execution. *Argument* within double quotation marks resolves while the DATA step is being constructed. For example, to invoke the macro SALES, you can use the following code:

```
call execute('%sales');
```

- The name of a DATA step character variable whose value is a text expression or a SAS statement to be generated. Do not enclose the name of the DATA step variable in quotation marks. For example, to use the value of the DATA step variable FINDOBS, which contains a SAS statement or text expression, you can use the following code:

```
call execute(findobs);
```

- -A character expression that is resolved by the DATA step to a macro text expression or a SAS statement. For example, to generate a macro invocation whose parameter is the value of the variable MONTH, you use the following code:

```
call execute('%sales('||month||')');
```

A Business Case

An analyst in a financial services organization (FSO) is interested in reading the firm's brokerage accounts data (stored as a flat file) into SAS for further modeling and analysis. The expectation is that there will be nearly 100 brokerage datasets each month which need to be loaded into SAS. The raw data is transmitted and received in a standard location (folder=N: /RAW_BROKERAGE/) on a daily basis. On the first day of each month, the analyst would like to load the previous month's files in full *that have not previously been loaded*.

Initial Load- Store Metadata

Below you will find a listing of files in the folder N:/RAW_BROKERAGE/ as of 1/2/13. The analyst will need to create a permanent SAS DATASET that will hold the names of the loaded files in a variable. This will be the initial metadata master.

Filename
FSO_Brokerage_20130101a.txt
FSO_Brokerage_20130101b.txt
FSO_Brokerage_20130101c.txt

Our first step is to load in all the above referenced files and create a permanent SAS dataset that houses the meta information on the files loaded (a listing of the files). In order to do this, we must first communicate to SAS about the files contained in the /RAW_BROKERAGE/ folder as of 1/2/13. In the SAS environment, the "x" command can be used to invoke a Linux command within a SAS program. In addition, we can use the SAS FILENAME statement, along with a PIPE command, to stream information via a LINUX invocation into a SAS FILENAME. Below, you will find SAS code that first uses the x command to point SAS to the /RAW_BROKERAGE/ folder via Linux, and then pipes in the contents of the folder to a FILENAME. Finally, we transform the piped in data after first using an infile statement.

```
libname brkg "N: /RAW_BROKERAGE/";
***linux command to point to folder;

x "cd /RAW_BROKERAGE/";

****pipes in contents of folder into file called "rawdata";

filename rawdata pipe "ls *";

***creates SAS dataset named "raw_brokerage" that contains two variables (1) Full: raw file name including .txt (2)
filename: variable created from substring of 'Full';

Data brkg.prev_loaded;
  Infile rawdata;
  Length full $27 filename $23;
  Input @1 full $ ;
  filename=substr(full, 1,23);
  if substr(filename, 5, 9 ) = 'Brokerage';
run;

proc print data= brkg.prev_loaded noobs;
title "Take A Look Inside brkg.prev_loaded";
run;
```

```
The PRINT Procedure

Full                                    filename
-----------------------------------------------------
FSO_Brokerage_20130101a.txt   FSO_Brokerage_20130101a
FSO_Brokerage_20130101b.txt   FSO_Brokerage_20130101b
FSO_Brokerage_20130101c.txt   FSO_Brokerage_20130101c
```

We now have a permanent SAS dataset `brkg.prev_loaded`, which contains the values of the files that we are going to read into SAS.

Loading with Call Execute

Now that we have a listing of the filenames in a permanent SAS dataset, we may proceed with the loading process. For the sake of this example, we will assume that the brokerage account only has six variables: `operations_date`, `cust_id`, `security_type`, `open_price`, `close_price`, `pct_chng`. In the code below, we will use a SAS MACRO to pass through two different parameters: `outfile` and `filename`. `Outfile` will be the name of the output SAS dataset, and `filename` will be the name of the input raw .txt file.

```
****macro to read in brokerage accounts;

%macro read_brokerage (outfile=, filename=);

        data &outfile;

         infile "N:/RAW_BROKERAGE/&filename"
         delimiter = "|"
         missover dsd lrecl=32767
         firstobs=2;

           INPUT
                OPERATIONS_DATE                  :YYMMDD8.
                CUST_ID                          :$10.
                SECURITY                         :$5.
                OPEN_PRICE                        :best30.2
                CLOSE_PRICE                       :best30.2
                PCT_CHNG                         :8.;

          format date date9.;
%mend read_brokerage;
```

We will run into one issue concerning our data loading operation. In the business case, we expect to have nearly one hundred different brokerage files to read into SAS. Using a SAS MACRO to load the files will help the analyst to avoid changing hard coded values through each iteration of the file read, **but** he/she will still have to write and edit nearly 100 macro calls!

To avoid this tedious process, we use the CALL EXECUTE routine. By passing the dataset containing the file metadata (to the CALL EXECUTE routine), we can create datastep code that will allow us to first specify a LIBNAME for an output location, then load a meta listing of files through our previously defined SAS MACRO ("%read_brokerage"). To do this, we will add the following block of code to the end of the SAS MACRO "%read_brokerage":

```
data _null_;
   set brkg.prev_loaded; /*master list of filenames*/

     ***specifies an output SAS LIBNAME ('BRKRG') using CALL EXECUTE***;
(1)  call execute("libname brkg 'N:/SAS_BROKERAGE/"||trim(substr(filename,14, 6))';");
(2)  call execute('read_brokerage(outfile= brkg.'||trim(filename)||', filename='||trim(filename)||'.txt); run;');
   run;
```

It is worth stepping through what happens in block of code above. The data set `brkg.prev_loaded` is read in during the data step. The variables 'full' and 'filename' are both contained in this dataset. 'Filename', if you remember, contains the names of the files in the /RAW_BROKERAGE/ folder as of 1/2/13. The dataset has three observations:

```
The SAS System

The PRINT Procedure

Full                                      filename
-----------------------------------------------------
FSO_Brokerage_20130101a.txt   FSO_Brokerage_20130101a
FSO_Brokerage_20130101b.txt   FSO_Brokerage_20130101b
FSO_Brokerage_20130101c.txt   FSO_Brokerage_20130101c
```

Stepping down to the line labeled (1), we use the CALL EXECUTE routine to specify the output LIBNAME. You will notice that we take a substring of the variable filename to create a portion of the libname path. The following will be the code generated by the CALL EXECUTE command in (1) for the first observation:

```
libname brkg 'N:/SAS_BROKERAGE/201301';
```

Now that the output library name has been specified using CALL EXECUTE, we can move down to (2). For the first observation (_n_=1) the following code will be generated by the call execute command:

```
%read_brokerage(outfile= brkg.FSO_Brokerage_20130101a, filename=FSO_Brokerage_20130101a.txt);
```

As you can see, the CALL EXECUTE routine outputs a macro call to our specifications. The parameter 'outfile' will give us the full LIBNAME and dataset name of the output data set, and the parameter 'filename' will be the name of the input text file that is read in using an INFILE statement. By using the CALL EXECUTE routine, we are able to step though the observations in `brkg.prev_loaded`, one at a time to create a dynamic output of executable SAS code. The above output code was only the output for _n_=1. For _n_=2 and _n_=3, the code generated by the CALL EXECUTE routine is as follows:

```
 libname brkg 'N:/SAS_BROKERAGE/201301';
%read_brokerage(outfile= brkg.FSO_Brokerage_20130101b, filename=FSO_Brokerage_20130101b.txt);
libname brkg 'N:/SAS_BROKERAGE/201301';
%read_brokerage(outfile= brkg.FSO_Brokerage_20130101c, filename=FSO_Brokerage_20130101c.txt);
```

Reconsider Our Business Case

We have succeeded in loading the brokerage accounts using a meta listing of files from N:/RAW_BROKERAGE/ as of 1/2/13. The CALL EXECUTE routine allowed us to dynamically create executable SAS code that specified the input file, output file and the output LIBNAME of the brokerage accounts we wished to load. We do not have to worry about changing the SAS code each time we load accounts.

We must now consider one additional problem in our brokerage ETL process. Imagine the loading analyst does not consider the brokerage data until 2/2/13. At this point he/she takes a look in the folder N:/RAW_BROKERAGE/, to find the following raw files:

Filename
FSO_Brokerage_20130101a.txt
FSO_Brokerage_20130101b.txt
FSO_Brokerage_20130101c.txt
FSO_Brokerage_20130102a.txt
FSO_Brokerage_20130102b.txt
FSO_Brokerage_20130102c.txt
FSO_Brokerage_20130103a.txt
FSO_Brokerage_20130103b.txt
FSO_Brokerage_20130103c.txt
. . .
FSO_Brokerage_20130129a.txt
FSO_Brokerage_20130129b.txt
FSO_Brokerage_20130129c.txt
FSO_Brokerage_20130130a.txt
FSO_Brokerage_20130130b.txt
FSO_Brokerage_20130130c.txt
FSO_Brokerage_20130131a.txt
FSO_Brokerage_20130131b.txt
FSO_Brokerage_20130131c.txt

You will notice a whole bunch of new files popped up since the last data load! There are three files per day, for each day of the month, for a total of 93 files. We have our SAS loading macro, along with our CALL EXECUTE code to create macro calls. We specify the correct output libraries from a meta file listing from N: /RAW_BROKERAGE/. There is just one small problem. We already loaded the first three files in this folder! We do not want to reload the same files over and over again as more raw data is transmitted to N:/RAW_BROKERAGE/. To solve this problem, we compare our list of already loaded files (`brkg.prev_loaded`), against the files currently in the folder. Any files that are found in both have already been loaded.

Our Solution

To avoid reloading all of the files in N:/RAW_BROKERAGE/ during each loading iteration, we must compare our previously loaded file dataset, `brkg.prev_loaded`, against a dataset containing the contents of

N:/RAW_BROKERAGE/ on 2/2/13. We recycle our code from the initial load (section 1) to create this dataset. The main difference will be the dataset we are creating will always house the full contents of N:/RAW_BROKERAGE/.

```
libname brkg "N: /RAW_BROKERAGE/";
 x "cd /RAW_BROKERAGE/";

 ****pipes in contents of folder into file called "rawdata";

 filename rawdata pipe "ls *";

         data brkg.full_contents;
          infile rawdata;
         length full $27 filename $23;
         input @1 full $ ;
         filename=substr(full, 1,23);
          if substr(filename, 5, 9 ) = 'Brokerage';
         run;

         proc print data= full_contents noobs;
          title "Take A Look Inside brkg.prev_loaded";
         run;
```

```
The SAS System

The PRINT Procedure

 Full                                          filename
 ----------------------------------------------------
 FSO_Brokerage_20130101a.txt   FSO_Brokerage_20130101a
 FSO_Brokerage_20130101b.txt   FSO_Brokerage_20130101b
 FSO_Brokerage_20130101c.txt   FSO_Brokerage_20130101c
 ...
 ...
 ...

 FSO_Brokerage_20130131a.txt   FSO_Brokerage_20130131a
 FSO_Brokerage_20130131b.txt   FSO_Brokerage_20130131b
 FSO_Brokerage_20130131c.txt   FSO_Brokerage_20130131c
```

The SAS dataset `brkg.full_contents` contains 93 obs (one for each file in N:/RAW_BROKERAGE/) and two variables. Now that we have our previously loaded files, and our current listing of files in N:/RAW_BROKERAGE/, we can compare the datasets to obtain the differences. The code below will match merge `brokerage.prev_loaded` with `brokerage.full_contents` and will only keep the files that have been added since the previous load.

```
 ***sort both datasets prior to match-merge***;
 proc sort data= brkg.full_contents; by filename; run;
 proc sort data= brkg.prev_loaded; by filename; run;


 ***merge datasets—only keep files in folder that havent been previously loaded***;
 data brokerage_loadnow;
  merge brkg.full_contents (in=x)
        brkg.prev_loaded (in=y);

     by filename;
     if x and not y;
 run;

 proc print data=brokerage_loadnow;
   TITLE "Brokerage Files to Load";
  var filename;
 run;
```

```
The SAS System

The PRINT Procedure

 filename
 --------------------------
 FSO_Brokerage_20130102a
 FSO_Brokerage_20130102b
 FSO_Brokerage_20130102c
 ...
 ...
 ...
```

```
FSO_Brokerage_20130131a
FSO_Brokerage_20130131b
FSO_Brokerage_20130131c
```

The SAS temporary dataset `brokerage_loadnow` contains a listing of all new files that have been transmitted to N:/RAW_BROKERAGE/ since the previous load. It contains 90 observations. Please take note that the three files loaded on 1/2/13 are not present in this dataset. We can now pass this dataset to the SAS MACRO facility for loading via the CALL EXECUTE routine (as previously demonstrated). The only difference is the dataset we are using the get our macro parameters.

```
data _null_;
   set brokerage_loadnow; /*master list of filenames*/

     ***specifies an output SAS LIBNAME ('brkg') using CALL EXECUTE***;
    call execute("libname brkg 'N:/SAS_BROKERAGE/"||trim(substr(filename,14, 6))';");
    call execute('%read_brokerage(outfile= brkg.'||trim(filename)||', filename='||trim(filename)||'.txt); run;');
run;
```

The CALL EXECUTE routine will dynamically create 90 macro calls for the macro '%read_brokerage (one for each of the files in the dataset `brokerage_loadnow`.

Update Metadata

A final step to ensure the right files are loaded on the next iteration is to update the metadata. We will update the `brkg.prev_loaded` dataset, so that when we compare the contents of N:/RAW_BROKERAGE/ against it next month, we aren't loading files that were already processed into SAS dataset during past loading iterations. We do this by appending the files loaded (`brokerage_loadnow`) to the previously loaded metadata listing (`brkg.prev_loaded`).

```
***add files just loaded to the 'previosly loaded' dataset for future processing***;
 Proc append base= brkg.prev_loaded data=brokerage_loadnow force;
 Run;
```

Comprehensive Code and Final Review

Below you will find the comprehensive code that will load new brokerage accounts from the N:/RAW_BROKERAGE/ folder. It is worthwhile to step through the process we used to get to this point.

1. We created a dataset `brkg.prev_loaded` that contained the filenames of the text files that were initially loaded out of N:/RAW_BROKERAGE/. These filenames were passed as a parameter through a SAS MACRO via the CALL EXECUTE routine.

2. We examined the N:/RAW_BROKERAGE/ folder about one month later and created another dataset `brkg.full_contents` which contains the full contents of the folder.

3. The `brkg.full_contents` dataset was compared against `brkg.prev_loaded` via a match merge to obtain the files that were transmitted to N:/RAW_BROKERAGE/ since the last load.

4. The dataset output from the match-merge, `brokerage_loadnow`, had its filename variable passed through the SAS loading MACRO via the CALL EXECUTE ROUTINE.

5. Finally, the filenames contained in the SAS DATASET `brokerage_loadnow` were appended to `brkg.prev_loaded` so the latter dataset is a representation of the most recently loaded files. Take note that the Day 1 process needs to only be run once. The main pupose of the Day 1 process is to obtain an initial metadata SAS dataset to append upon, and to fine tune the loading SAS MACRO.

Going forward the loading analyst can run the day 2 code noted below, without worrying about reloading datasets. SAS will dynamically build all of the macro calls needed to load any new brokerage files transmitted. Only files that are new to the N:/RAW_BROKERAGE/ folder will be loaded, as a metadata file containing the names of previously loaded files keeps track of these incremental changes. Countless man-hours will be saved by (1) allowing SAS to perform otherwise tedious cross-checking of files to be loaded and, (2) Allowing SAS to build macro calls dynamically.

```
***COMPREHENSIVE CODE*****;

/*DAY 1 (run just ONCE): CREATES A LISTING OF FILES LOADED, USES CALL EXECUTE TO ACCESS LOADING MACRO*/

        Libname brkg "N: /RAW_BROKERAGE/";
        ***linux command to point to folder;

        x "cd /RAW_BROKERAGE/";

        ****pipes in contents of folder into file called "rawdata";

        filename rawdata pipe "ls *";

        ***creates SAS dataset named "raw_brokerage" that contains two variables (1) Full: raw file name including .txt (2)
        filename: variable created from substring of 'Full';

        Data brkg.prev_loaded;
          Infile rawdata;
          Length full $27 filename $23;
          Input @1 full $ ;
          filename=substr(full, 1,23);
          if substr(filename, 5, 9 ) = 'Brokerage';
        run;


        %macro read_brokerage (outfile=, filename=);

                 data &outfile;

               infile "N:/RAW_BROKERAGE/&filename"
               delimiter = "|"
               missover dsd lrecl=32767
               firstobs=2;

               INPUT
                         OPERATIONS_DATE                  :YYMMDD8.
                         CUST_ID                           :$10.
                         SECURITY                         :$5.
                         OPEN_PRICE                       :best30.2
                         CLOSE_PRICE                       :best30.2
                         PCT_CHNG                          :8.;

                   format date date9.;
        %mend read_brokerage;

       data _null_;
           set brkg.prev_loaded; /*master list of filenames*/

             ***specifies an output SAS LIBNAME ('brkg') using CALL EXECUTE***;
            call execute("libname brkg 'N:/SAS_BROKERAGE/"||trim(substr(filename,14, 6))';");
            call execute('read_brokerage(outfile= brkg.'||trim(filename)||', filename='||trim(filename)||'.txt); run;');
       run;



/*DAY 2: CREATES A LISTING OF FILES CURRENTLY IN FOLDER, AND SENDS FILENAME AS PARAMETER THROUGH SAS MACRO VIA CALL EXECUTE
ROUTINE.  THIS IS THE PROGRAM THAT WILL BE RUN TO LOAD PROGRAMS ON A MONTHLY BASIS****/

        libname brkg "N: /RAW_BROKERAGE/";

        x "cd /RAW_BROKERAGE/";

        ****pipes in contents of folder into file called "rawdata";

        filename rawdata pipe "ls *";

        data brkg.full_contents;
                  Infile rawdata;
                 Length full $27 filename $23;
                 Input @1 full $ ;
                 filename=substr(full, 1,23);
                  if substr(filename, 5, 9 ) = 'Brokerage';
                 run;

                 proc print data= full_contents noobs;
```

```
          title "Take A Look Inside brokerage.prev_loaded";
         run;

***sort both datasets prior to match-merge***;
proc sort data= brkg.full_contents; by filename; run;
proc sort data= brkg.prev_loaded; by filename; run;


***merge datasets—only keep files in folder that havent been previously loaded***;
data brokerage_loadnow;
 merge brkg.full_contents (in=x)
       brkg.prev_loaded (in=y);

    by filename;
    if x and not y;
run;


%macro read_brokerage (outfile=, filename=);


         data &outfile;

          infile "N:/RAW_BROKERAGE/&filename"
        delimiter = "|"
        missover dsd lrecl=32767
        firstobs=2;

        INPUT
                  OPERATIONS_DATE                  :YYMMDD8.
                  CUST_ID                    :$10.
                  SECURITY                   :$5.
                  OPEN_PRICE                        :best30.2
                  CLOSE_PRICE                :best30.2
                  PCT_CHNG                         :8.;

           format date date9.;
%mend read_brokerage;

data _null_;
    set brokerage_loadnow; /*master list of filenames*/

      ***specifies an output SAS LIBNAME ('brkg') using CALL EXECUTE***;
    call execute("libname brkg 'N:/SAS_BROKERAGE/"||trim(substr(filename,14, 6))';");
    call execute('%read_brokerage(outfile= brkg.'||trim(filename)||', filename='||trim(filename)||'.txt); run;');
 run;

***add files just loaded to the 'previosly loaded' dataset for future processing***;
   Proc append base= brkg.prev_loaded data=brokerage_loadnow force;
   Run;
```

REFERENCES

1. H. Ian Whitlock (1997). 'Call Execute: How and Why'. *Proceedings of the Twenty Second Annual SAS Users Group International Conference*. San Diego, CA. pp. 410-414.
2. SAS Institute Inc., SAS ® 9.2 Macro Language Reference. 'Call Execute Routine'. Cary, NC.
3. Sharma, Rajkumar. 'Call Execute: A Hidden Treasure and a Powerful Function'. South San Francisco, CA.