

Custom SAS Excel® Toolbox

Michael McCarthy PE, Austin Energy, Austin TX USA

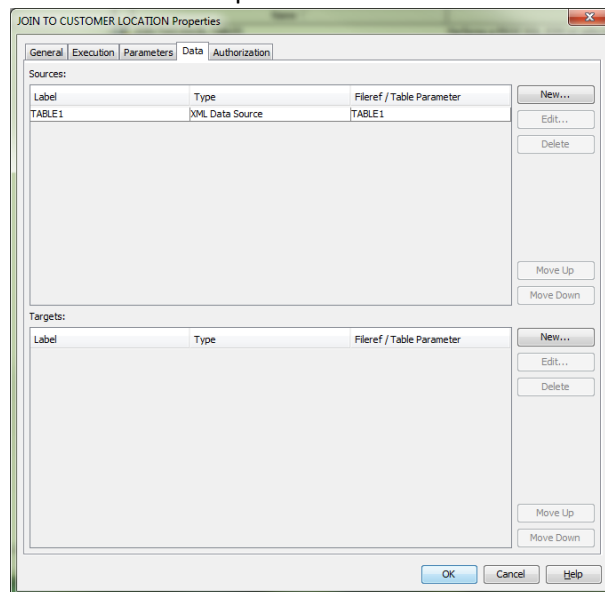
ABSTRACT

Data users frequently have a need to merge Microsoft Excel® data with other data sets. While the TASK tab within the SAS Add-In 7.12 for Microsoft Office® ribbon offers access to a variety of SAS analytics, it doesn't provide an option to join or merge user-selected cells. This paper is written for BASE SAS® programmers who want to use the SAS Add-In 7.12 for Microsoft Office® to perform a "PROC SQL JOIN" on user-selected Microsoft Excel® data, without leaving the Excel® environment.

INTRODUCTION

The SAS Add-In 7.12 for Microsoft Excel® is a powerful interface to the BASE SAS® engine. Selected Excel® data can be read into BASE SAS® as XML through a menu selection, processed on the SAS® Workspace server, and then sent back to the users' Excel® workbook. The trick is to define the selected Excel® table as an "XML data source" within the "Data" tab in the Stored Process Properties as shown below:

Stored Process Properties Window:



Defining a data source as XML, or "Extensible Markup Language", allows a stored process to reference the selected Excel® data using the **LIBNAME** statement. Specifically, within the script this becomes

LIBNAME <SOURCE> xml;

When defined as a library, the actual XML data can be loaded into a SAS Table using the reference "<SOURCE>.&_webin_sasname", where "&_webin_sasname" is a reserved name for the SAS data associated with the XML source.

Each selected XML table will have its own a “KEY” field to join on, as well as a selection of headers and corresponding data. Since the selected KEY cells may contain several separate words, it’s important to use the **TRANWRD** command to replace blank cells with “_”. Also keep in mind that the column data for each selected KEY fields could describe text or numeric data, so the context will need to be known before performing a **JOIN** between the data tables.

A simple way to resolve the context of the KEY fields within our JOIN is to force them into a character format with a **CAT** function. The CAT function will add any character we specify to the input and output a character values, regardless of input type. While this is a fairly robust solution when a user selects tables from the same spreadsheet, it could cause problems when joining data to standard SAS® data sets.

A more robust solution is to utilize the **VARNUM** and **VARTYPE** functions to determine the format for a selected KEY. The VARNUM statement is helpful since it will bring back the column position within the selected data table. Once the column position is known, then the VARTYPE function can recover the format type for the enumerated column, where the function returns “C” for character or “N” for number. Thus, it’s possible to use this information to select between macros that join on either numeric or character data KEY fields.

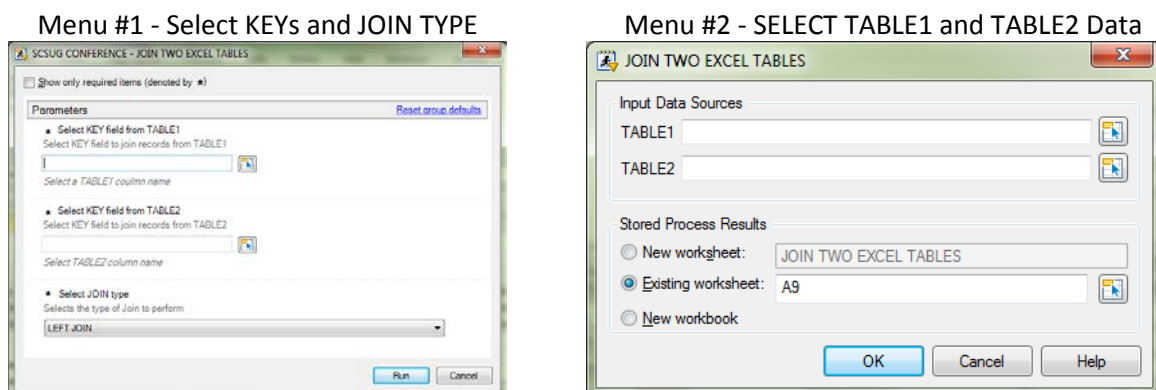
With the formatting for both KEYS defined or determined as previously discussed, the various types of JOIN functions are performed within the context of a **MACRO**. The types of JOIN functions presented within scripts discussed are: **LEFT JOIN**, **RIGHT JOIN**, **INNER JOIN**, **CROSS JOIN**, **INTERSECT**, **EXCEPT ALL**, **UNION ALL CORRESPONDING**.

METHODOLOGY

The BASE SAS® scripts shown in the Appendix I and II are stored in the metadata and execute on the Workspace server. These can be accessed from the Stored Process Web Application, Enterprise Guide®, or Microsoft Office®.

SCRIPT OVERVIEW: Appendix I - JOIN TWO USER SELECTED EXCEL® TABLES

When the user selects the report link, the following prompt menus will appear (MS Excel® version shown):



Upon opening the stored process, the user is prompted with the Menu #1 screen to select a “KEY” from each of the two tables to be joined, as well as the type of “JOIN”. The prompt values are processed within the stored process as the corresponding macro variables: “&KEY1”, “&KEY2”, and “&JOIN”. Here, “&KEY1” and “&KEY2” are selected individual cells, and “&JOIN” is a selection from a drop down list (text selection from a static list).

The user highlights the cells within TABLE1 and TABLE2 on the Menu #2 screen, making sure to include the table column labels. These are defined as XML data sources as previously discussed, and can be referenced within our script as “TABLE1.&_webin_sasname” and “TABLE2.&_webin_sasname”.

BASE SAS®: Appendix I - JOIN TWO USER SELECTED EXCEL® TABLES

Once the search information has been collected, the stored process begins by loading the macros “JOIN”, “XJOIN”, and “UJOIN”, followed by the LIBNAME XML definitions of TABLE1 and TABLE2 (Appendix I, Steps 1-15). These XML data sources are then loaded into the corresponding SAS® data sets INPUT1 and INPUT2, while the “&KEY1” and “&KEY2” selections are placed in global variables “CHKKEY1” and “CHKKEY2” (Appendix I, steps 16 - 19). Here, the “call symput” command is used with the TRANWRD function to remove blanks from the KEY fields. With all these user selections complete, the selected JOIN Macro is called, and a JOIN is performed using the CAT function to force a character comparison of the KEYS. In particular,

Appendix I, Step 4: “... CAT('A',a.&Field1) = CAT('A',b.&Field2)...” /*Force a string comparison*/

Where macro variables &Field1 and &Field2 above have “blank free” values from global variables “&CHKKEY1” and “&CHKKEY2” (Appendix II – Steps 17 and 19).

Once the join is complete, the output tables “INPUT1”, “INPUT2” and “JOINED” are then sent back to the user’s Excel® workbook (Appendix I, steps 24 to 26).

SCRIPT OVERVIEW: APPENDIX II – JOIN A SELECTED EXCEL® TABLE TO A SAS® DATA TABLE

When the user selects the report link, the prompt menus will appear (MS Excel® version shown):

Menu #1 - Select KEYS and JOIN TYPE

JOIN TO PROPERTY TAX INFO

☐ Show only required items (denoted by ★)

★ Select KEY field from TABLE1
Select KEY field to join records from TABLE1
(Reference: [Join Test.xlsx]Sheet2!A3)
Test
Select a TABLE1 column name

★ Select KEY field from TABLE2
Select KEY field to join records from TABLE2
PROPERTYID [TCADWCAD(Table)]

★ Select JOIN type
Selects the type of Join to perform
LEFT JOIN

★ Data Warehouse Library
Data Warehouse Library
/Shared Data/TCADWCAD(Library) TCADWCAD X Browse...

★ TCAD and WCAD Information
TCAD and WCAD Information
TCADWCAD is a table containing all Property Tax information for Travis and Williamson Counties
/Shared Data/TCADWCAD(Table) Browse...

Run Cancel

Menu #2 - SELECT TABLE1 and TABLE2 Data

JOIN TO PROPERTY TAX INFO

Input Data Sources
TABLE1 [Join Test.xlsx]Sheet2!\$A\$3:\$A\$5

Stored Process Results
☒ New worksheet: JOIN TO PROPERTY TAX INFO
☐ Existing worksheet:
☐ New workbook

OK Cancel Help

Upon opening the stored process, the user is prompted with the Menu #1 screen to select a KEY from the Excel® table, a corresponding KEY from an external SAS® data table, and the type of JOIN to perform. The prompt values are processed within the stored process as macro variables: “&KEY1”, “&KEY2”, and “&JOIN”. Here, “&KEY1” is selected individual cell, “&KEY2” is a read only data source prompt, and “&JOIN” is a selection from a drop down list (text selection from a static list). Note that a read only data library and Base SAS® data set are also defined on this menu. Within the script, these are referred to as “&LIBRARY” and “&TABLE2”.

On the Menu #2 screen, the user highlights the cells within TABLE1, including the table column labels. This is referenced within our script as “TABLE1.&_webin_sasname”. The selected cells will be joined to the existing data table: “&LIBRARY.&TABLE2” using the selected KEY. As in Script I, the TRANWRD function is used to remove blanks from the selected KEY.

BASE SAS®: APPENDIX II – JOIN A SELECTED EXCEL® TABLE TO A SAS® DATA TABLE

Script II is fundamentally almost identical to Script I. The only real difference is that Script II will join a selected Excel® table to an existing SAS® table. To facilitate this, both numeric and character versions of &KEY1 are defined within a data step that defines INPUT1 (Appendix II, Step 12). Here “KEY1TXT” represents the value in a character format, while “KEY1NUM” is a numeric format. The appropriate format to use within the JOIN will depend upon KEY2, which has a known format determined within the following snippet (Appendix II, Step 15):

Code	Purpose
<pre>/*15*/ %let dsid=%sysfunc(open(&INPUT2,i));</pre>	-> Opens selected SAS® library table
<pre>%let Num=%sysfunc(varnum(&dsid,&KEY2));</pre>	-> Finds the numeric column position of the KEY2
<pre>%let TYPE=%sysfunc(vartype(&dsid,&Num));</pre>	-> Use table column number to find field format

The value of “TYPE” (or &TYPE) is either ‘C’ or ‘N’. This is used to select an appropriate JOIN (Appendix II, Steps 16 – 18). For a character format, “JOINTXT” is called, while a numeric format will call “JOINNUM”.

Once the join is complete, the output tables “INPUT1” and “JOINED” are then sent back to the user’s Excel® workbook (Appendix II, steps 19 and 20).

CONCLUSION

The BASE SAS® code presented in this paper is used within the SAS Add-In 7.12 for Microsoft Office® to perform a join on user selected Microsoft Excel® data. The macro presented in Appendix I demonstrates a basic script to join two user selected Excel tables, while the script shown in Appendix II shows how a selected Excel table can be joined with existing SAS data tables. Using stored processes within Excel® makes SAS® more accessible, empowering users to conveniently join or merge data without leaving the Microsoft Excel® environment.

APPENDIX I – JOIN TWO USER SELECTED EXCEL® TABLES

```
/******  
/* JOIN TWO USER SELECTED EXCEL® TABLES  
/* WRITTEN BY MIKE MCCARTHY, PE  
/* DATE: 10/24/2016  
/* MERGE TWO SOURCE TABLES AND OUTPUT AS EXCEL®  
/******  
/*STEP*/  
/*1 */ %STPBEGIN;  
  
/*Macro to perform LEFT, RIGHT, INNER JOINS*/  
/*2 */ %macro JOIN(INDIR,INTABLE1,INTABLE2,OUTDIR,OUTTABLE,FIELD1,FIELD2,JOIN);  
/*3 */ proc sql dquote=ansi;  
        create table &OUTDIR.&OUTTABLE (DROP = &Field1) as  
        select distinct a.&Field1 as TABLE1KEY,&INTABLE1..*,b.&Field2 as TABLE2KEY,&INTABLE2..*  
/*4 */ from &INDIR.&INTABLE1 as a &JOIN &INDIR.&INTABLE2 as b on CAT('A',a.&Field1) = CAT('A',b.&Field2)  
        order by TABLE1KEY;  
        quit;  
/*5 */ %mend JOIN;  
  
/*Macro to perform CROSS JOIN*/  
/*6 */ %macro XJOIN(INDIR,INTABLE1,INTABLE2,OUTDIR,OUTTABLE,FIELD1,FIELD2,JOIN);  
/*7 */ proc sql dquote=ansi;  
        create table &OUTDIR.&OUTTABLE (DROP = &Field1) as  
/*8 */ select distinct a.&Field1 as TABLE1KEY,&INTABLE1..*,b.&Field2 as TABLE2KEY,&INTABLE2..*  
        from &INDIR.&INTABLE1 as a &JOIN &INDIR.&INTABLE2 as b;  
        quit;  
/*10*/ %mend XJOIN;  
  
/*Macro to perform INTERSECT, EXCEPT ALL, UNION ALL CORRESPONDING*/  
/*11*/ %macro UJOIN(INDIR,INTABLE1,INTABLE2,OUTDIR,OUTTABLE,FIELD1,FIELD2,JOIN);  
/*12*/ proc sql dquote=ansi;  
        create table &OUTDIR.&OUTTABLE as  
        select *  
        from &INDIR.&INTABLE1  
        &JOIN  
        select *  
        from &INDIR.&INTABLE2;  
        quit;  
/*13*/ %mend UJOIN;  
  
/*Define the Selected Tables - The Key selection is defined using input parameters*/  
/*14*/ LIBNAME TABLE1 xml; /*The global variable '&_webin_sasname' is reserved for the input table names*/  
/*15*/ LIBNAME TABLE2 xml; /*The global variable '&_webin_sasname' is reserved for the input table names*/
```

```

/*16*/ data work.INPUT1;
      set TABLE1.&_webin_sasname;
/*17*/ call symput('CHKKEY1', TRANWRD("&KEY1", " ", "_"));/*Create global variable for KEY without spaces */
      run;

/*18*/ data work.INPUT2;
      set TABLE2.&_webin_sasname;
/*19*/ call symput('CHKKEY2', TRANWRD("&KEY2", " ", "_"));/*Create global variable for KEY without spaces*/
      run;

/*20*/ data _NULL_;
/*21*/ if "&JOIN" in ('LEFT JOIN','RIGHT JOIN','INNER JOIN') then Call Execute
      ('%JOIN(WORK,INPUT1,INPUT2,WORK,JOINED,&CHKKEY1,&CHKKEY2,&JOIN)');

/*22*/ if "&JOIN" in ('CROSS JOIN') then Call Execute
      ('%XJOIN(WORK,INPUT1,INPUT2,WORK,JOINED,&CHKKEY1,&CHKKEY2,&JOIN)');

/*23*/ if "&JOIN" in ('INTERSECT','EXCEPT ALL','UNION ALL CORRESPONDING') then Call Execute
      ('%UJOIN(WORK,INPUT1,INPUT2,WORK,JOINED,&CHKKEY1,&CHKKEY2,&JOIN)');
      run;

/*24*/ proc print data=INPUT1 noobs;
      run;
/*25*/ proc print data=INPUT2 noobs;
      run;
/*26*/ proc print data=JOINED noobs;
      run;
%STPEND;/*END STORED PROCESS*/

```

APPENDIX II – JOIN A SELECTED EXCEL® TABLE TO A SAS® DATA TABLE

```
/******  
/* JOIN A USER SELECTED EXCEL® TABLE TO A SAS DATA TABLE      */  
/* WRITTEN BY MIKE MCCARTHY, PE                                     */  
/* DATE: 10/24/2016                                                */  
/* MERGE TWO SOURCE TABLES AND OUTPUT AS EXCEL®                 */  
/******  
/*STEP*/  
/*1 */ %STPBEGIN;  
/******  
/*          JOINTX and JOINNUM Macro                               */  
/* MERGE TWO SOURCE TABLES AND OUTPUT                           */  
/* INDIR1, INDIR2 = Source Directories for                         */  
/* TABLE1, TABLE2 = Source Tables                               */  
/* OUTDIR = Output Directory for Joined Table                     */  
/* FIELD1, FIELD2 = Key Fields used to perform join               */  
/* JOIN = Type of Join to perform                                 */  
/******  
/*JOINTX MACRO*/  
/*2 */ %macro JOINTXT(INDIR1,INTABLE1,INDIR2,INTABLE2,OUTDIR,OUTTABLE,FIELD1,FIELD2,JOIN);  
/*3 */ proc sql dquote=ansi;  
        create table &OUTDIR.&OUTTABLE (DROP = &Field1 KEY1NUM KEY1TXT) as  
        select distinct a.&Field1 as TABLE1KEY,&INTABLE1..*,b.&Field2 as TABLE2KEY,&INTABLE2..*  
/*4 */ from WORK.INPUT1 as a &JOIN &INDIR2.&INTABLE2 as b on (a.KEY1TXT = INPUT(b.&KEY2, $50.) or  
        a.KEY1NUM = INPUT(b.&KEY2, 10.) and not missing(KEY1NUM) )/*Avoid Missing KEY1NUM*/  
        order by a.KEY1TXT;  
        quit;  
/*5 */ %mend JOINTXT;  
  
/*JOINNUM MACRO*/  
/*6 */ %macro JOINNUM(INDIR1,INTABLE1,INDIR2,INTABLE2,OUTDIR,OUTTABLE,FIELD1,FIELD2,JOIN);  
/*7 */ proc sql dquote=ansi;  
        create table &OUTDIR.&OUTTABLE (DROP = &Field1 KEY1NUM KEY1TXT) as  
        select distinct a.&Field1 as TABLE1KEY,&INTABLE1..*,b.&Field2 as TABLE2KEY,&INTABLE2..*  
/*8 */ from WORK.INPUT1 as a &JOIN &INDIR2.&INTABLE2 as b on (a.KEY1NUM = b.&KEY2 and not  
        missing(KEY1NUM) ) /* Avoid Missing KEY1NUM */  
        order by a.KEY1TXT;  
        quit;  
/*9 */ %mend JOINNUM;  
  
/*10*/ LIBNAME TABLE1 xml; /*Define all the available Libraries within the script*/  
        LIBNAME TCADWCAD '/media/ssd1/SAS.PROD/PROD/TCADWCAD/';  
        LIBNAME CUSTOMER '/media/ssd1/SAS.PROD/PROD/CUSTOMER/';  
        LIBNAME FINANCE '/media/ssd1/SAS.PROD/PROD/FINANCE/';
```

```

/*11*/ data work.INPUT1; /*Define the initial table to replace blanks within the field names by " _"*/
      set TABLE1.&_webin_sasname; /*The global variable '&_webin_sasname' is reserved */
      call symput('KEYCHECK', TRANWRD("&KEY1", " ", "_"));
      run;

/*12*/ data work.INPUT1; /*Create Text and Number versions of the input variable*/
      set work.INPUT1;
      format KEY1TXT $50.;
      format KEY1NUM 12.;
      KEY1TXT = &KEYCHECK; /*Replace Blanks with ' _'*/
      KEY1NUM = &KEYCHECK; /*Replace Blanks with ' _'*/
      run;

/*13*/ data _null_;
      call symput('GET', scan("&TABLE2", 3)); /*Find the Library Data Table*/
      run;

/*14*/ data _null_;
      call symput('INPUT2', "&LIBRARY..&GET"); /*Identify the Library.Name*/
      run;

/*Script to find the position and variable type of user selected KEY2*/
/*15*/ %let dsid=%sysfunc(open(&INPUT2,i)); /*Open SAS data table*/
      %let Num=%sysfunc(varnum(&dsid,&KEY2)); /*Find the numeric position of KEY2*/
      %let TYPE=%sysfunc(vartype(&dsid,&Num)); /*Find the type, C or N, of this value*/

/*16*/ data _null_;
      INDIR2 = scan("&INPUT2", 1); /*Parse the directory*/
      INTAB2 = scan("&INPUT2", 2); /*Parse the table name*/
      TYPE = "&Type"; /*Identify the type of the variable C or N, Character or Number*/
/*17*/ if "&JOIN" in ('LEFT JOIN', 'INNER JOIN') and Type = 'N' then Call Execute
      ('%JOINNUM(WORK, INPUT1, &LIBRARY, &GET, WORK, JOINED, KEY1TXT, &KEY2, &JOIN)');
/*18*/ if "&JOIN" in ('LEFT JOIN', 'INNER JOIN') and Type = 'C' then Call Execute
      ('%JOINTXT(WORK, INPUT1, &LIBRARY, &GET, WORK, JOINED, KEY1TXT, &KEY2, &JOIN)');
      run;

/*19*/ proc print data=INPUT1 (DROP = KEY1TXT KEY1NUM) noobs;
      run;
/*20*/ proc print data=JOINED noobs;
      run;

%STPEND;

```

ACKNOWLEDGEMENTS

Special thanks to John Trowbridge, PE (DABI Engineering Supervisor, Austin Energy), Liz Jambor, DABI Manager, Austin Energy), and Debbie Kimberly (VP Customer Energy Solutions, Austin Energy) for their continued support of our SAS Data Analytics Server.

REFERENCES

Paper 233-30 An Introduction to SAS® Character Functions (Including Some New SAS®9 Functions)

Ronald Cody, Ed.D

<http://www2.sas.com/proceedings/sugi30/233-30.pdf>

SAS(R) 9.3 XML LIBNAME Engine: User's Guide, Second Edition

<https://support.sas.com/documentation/cdl/en/engxml/62845/HTML/default/viewer.htm#titlepage.htm>

Quick Hits – My Favorite SAS® Tricks

Marje Frecht

<http://analytics.ncsu.edu/sesug/2012/IT-03.pdf>

Address Cleaning Using the TRANWRD Function

Rena Jones, Mike Zdeb

<http://www.lexjansen.com/nesug/nesug08/cc/cc07.pdf>

%SYSFUNC – The Brave New Macro World

Chris Yindra, C. Y. Training Associates

<http://www2.sas.com/proceedings/sugi23/Advtutor/p44.pdf>

I value your comments. You can contact the author at:

Michael McCarthy, PE

Austin Energy, Customer Energy Solutions, Austin TX USA

michael.mccarthy@austinenergy.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.