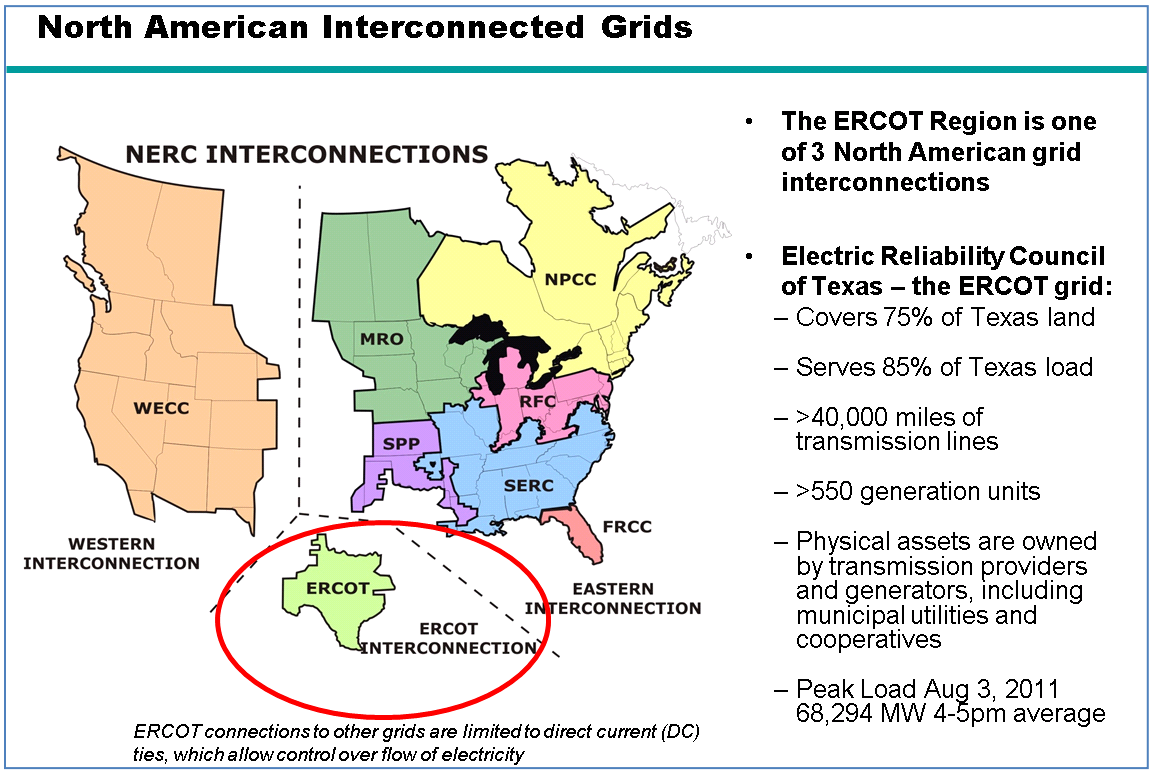
Using SAS to Read From and Write to EXCEL Workbooks Set Up as Templates Which Are Not Set Up In Columns

Carl L Raish, Electric Reliability Council of Texas (ERCOT)

BACKGROUND INDORMATION

This paper will review some sample SAS code developed at to exchange information with some entities in the ERCOT market. By way of providing some context to the code, ERCOT oversees the operations of a major portion of the electric grid in Texas. Our functions include managing reliability of the grid by procuring ancillary services to ensure a continuous balance between generation and load.



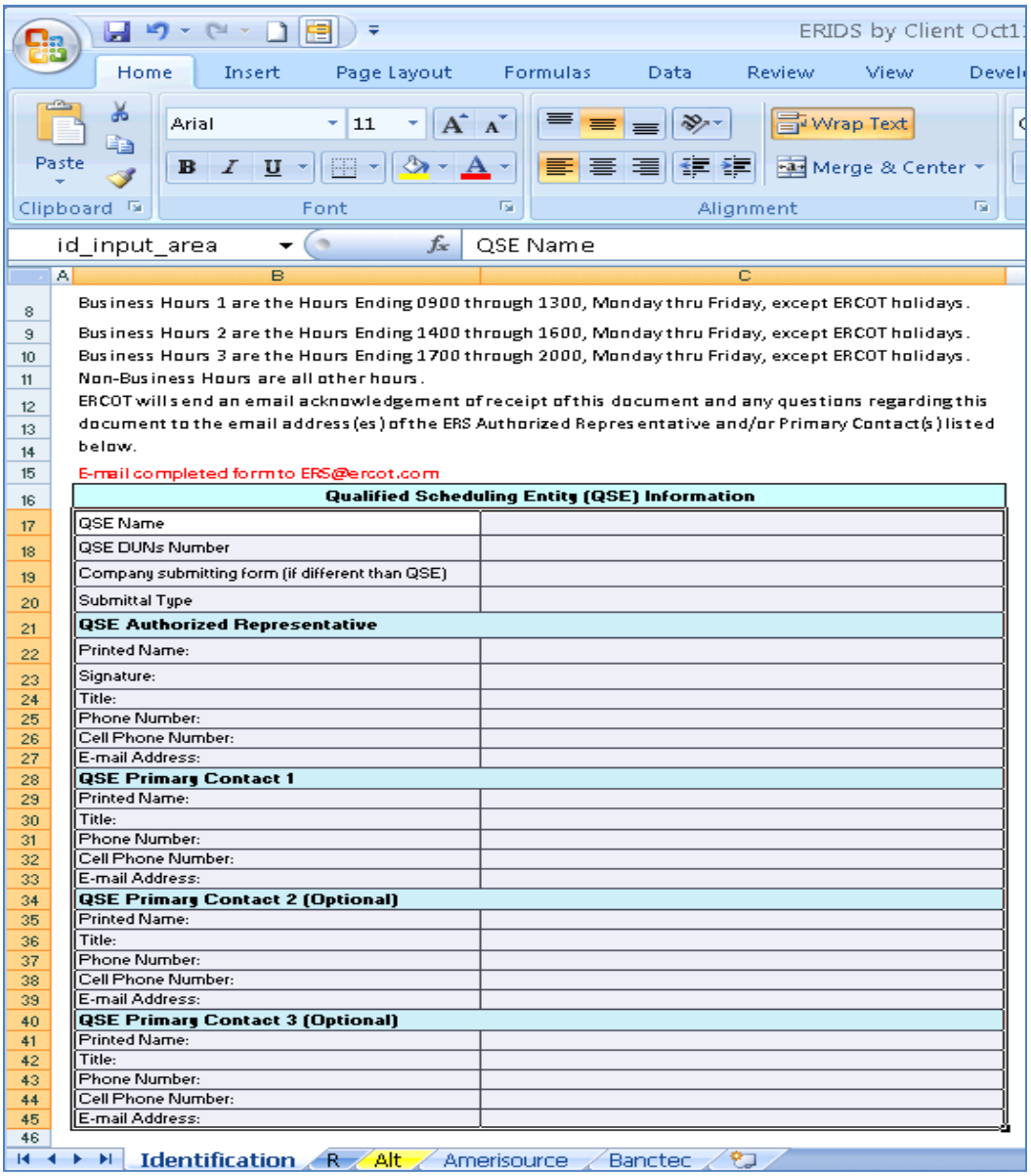
Among the services ERCOT procures is the Emergency Response Service; under the terms of this service electricity users agree to reduce their consumption in the late stages of a grid emergency. ERCOT has a $50 million annual budget to pay those users for committing to quickly reduce their electricity consumption; and we run auctions three times a year to buy the service. The procurement is run in two stages: a resource identification phase and an offer phase.

ERCOT has established an Excel form as the mechanism for users (actually a Qualified Scheduling Entity … QSE … representing those users) to identify customers potentially interested in providing the service. ERCOT processes the form with SAS, and returns the form with diagnostic and other information to assist the QSE in submitting an offer. This completes the resource identification phase.

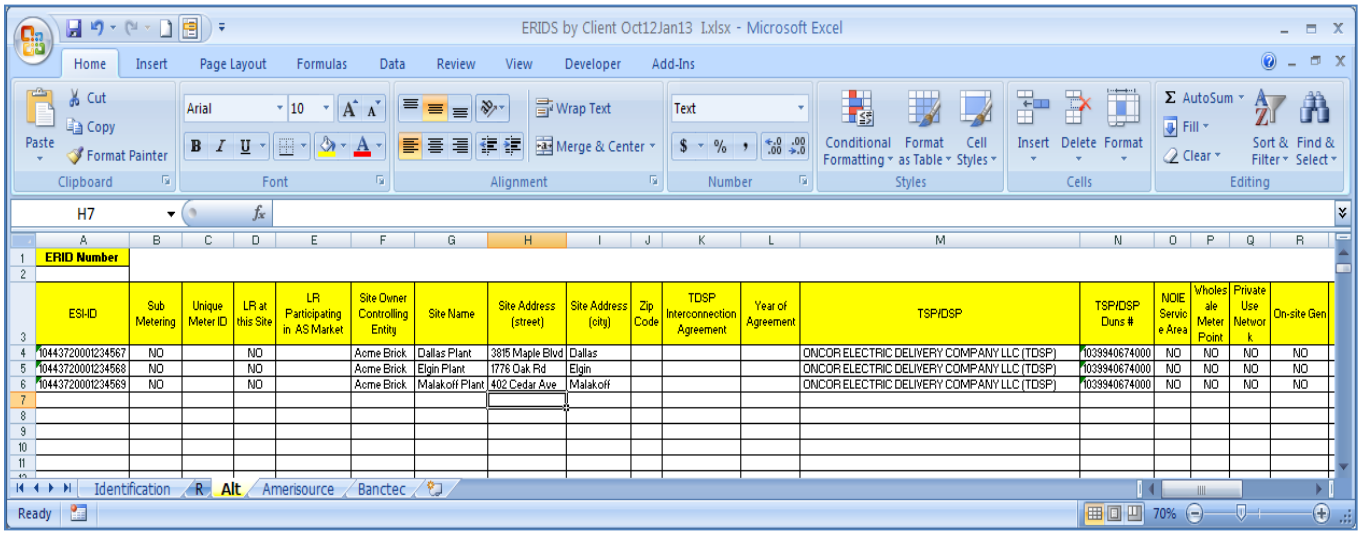
In the offer phase, the QSE fills in additional fields on the form (especially including prices) and returns the form to ERCOT.

The Excel form, pictured below, has three different types of tabs: Identification, Alt and other arbitrarily named tabs. The Identification tab is used for identifying the QSE submitting the form and a set of contact information for the QSE. The tab is set up with a named range, 'id_input_area' to facilitate reading just the required fields with SAS. Cells outside of the range contain information that the QSE should be aware of when submitting the form, but which is not needed for processing the form.

A special version number cell is set up on the tab which is formatted with a white text color to make it less obvious. The version number is changed whenever the form is modified and is used to validate that the QSE is submitting the current form. Since the workbook is being processed with SAS, this is critical to ensure that the data required from the form is pulled from the correct cells. The version number is set up on all tabs in the workbook.



The ‘Alt’ tab provides a way for the QSE, during the resource identification phase, to identify individual site participants that may subsequently be grouped together into an aggregation of sites as a resource. The name ‘Alt’ must be used, and, as a result, there can only be one Alt tab in the workbook. If the QSE chooses not to use the tab, it may be either deleted from the workbook or simply left blank. No named ranges are set up on the ‘Alt’ tab; the version number cell is set up as described above for the Identification tab, and a cell is reserved in A2, to be filled in by ERCOT as a control number to be associated with submitted participants in all subsequent processing for the procurement.



The other tabs, American and Bankers Inc, are filled in as resources. The ‘R’ tab is a pre-formatted blank tab provided to facilitate the addition of tabs by the QSE. The tab names are arbitrarily assigned by the QSE, and some of the cells are set up as drop down lists. The cells referred to for the drop down list are off to the side of the form and are formatted with a white color to make them less subject to tampering. No named ranges are set up on the these tabs; the version number cell is set up as described above for the Identification tab, and a cell is reserved in A2, to be filled in by ERCOT as a control number to be associated with the specific resource in any subsequent processing. The baseline options for the drop down list associated with cell C4 are set on the standard form posted for use at the beginning of the resource identification process and are filled in by SAS when outputting the approved form based on the baseline analysis performed by ERCOT. These cells are also formatted with a white text color to make them less obvious.

ERIDS by Client Oct12Jan13 1.xlsx - Microsoft Excel

ERID Number	Renewal Opt In	Resource Type	ERS Time Period	Include Time	Declared Self-Serve	Declared Injection	Capacity Offer (MW)	Price ($/MW/hr)	Maximum Base	MW Tied Offers
		Load	Business Hours 1	Yes						
Resource Name		Baseline	Business Hours 2	Yes						
American Eagle			Business Hours 3	Yes						
			Non Business Ho	Yes						

ESI-ID	Sub Metering	Unique Meter ID	LR at this Site	LR Participating	Site Owner Controlling Entity	Site Name	Site Address (street)	Site Address (city)	Zip Code	TDSP Interconnection Agreement	Year of Agreement	TSP/DSP	TSP/DSP Duns #	S
	YES	00115164	NO		American	Eagle	3101 Gaylord Parkw	Frisco	75034			BRAZOS ELECTRIC POWER CO OP INC (TDSP)	0037722904000	YES

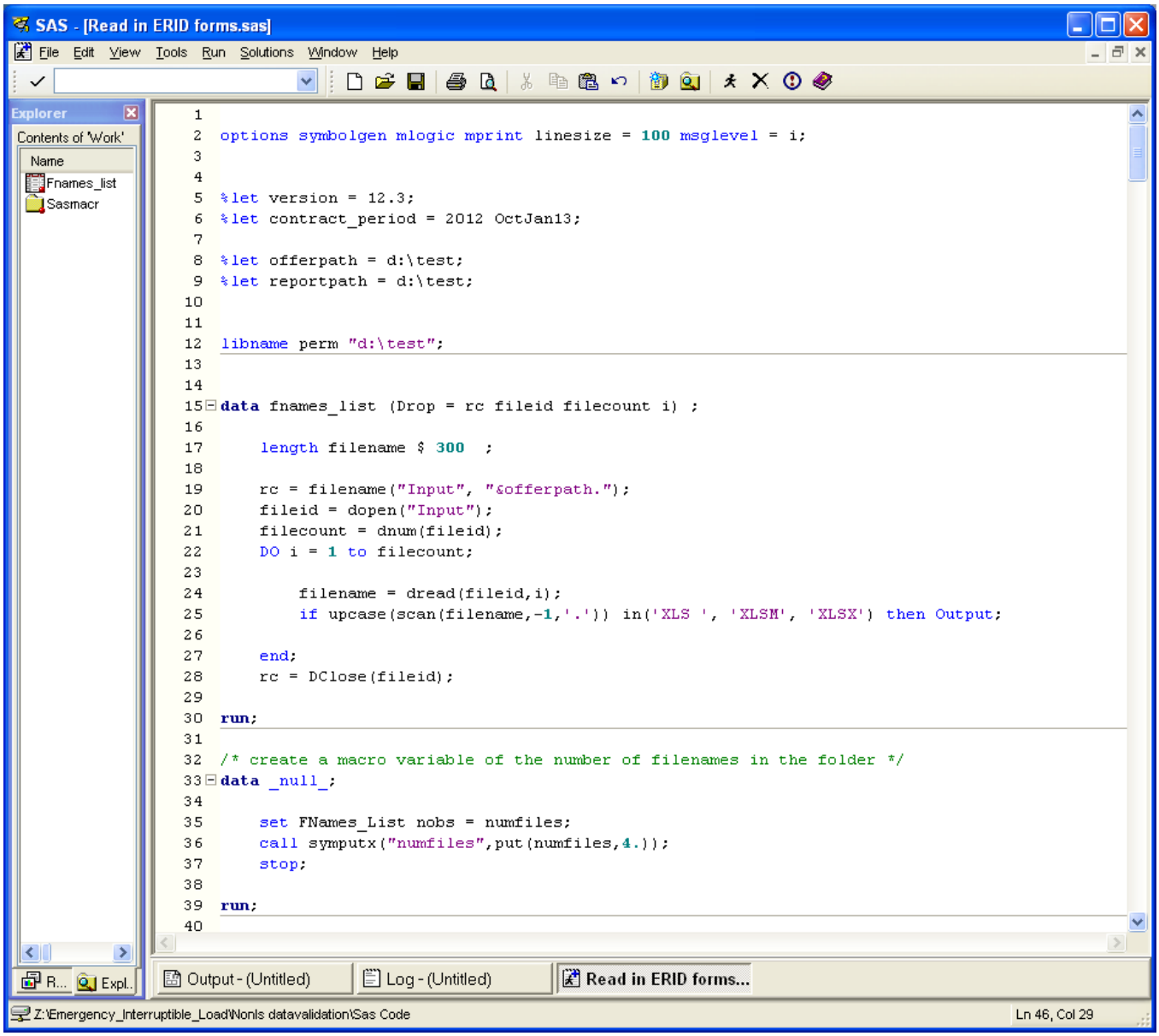
Identification | R | Alt | American Eagle | Bankers Inc

Ready 70%

READING THE SUBMISSION FORM WITH SAS

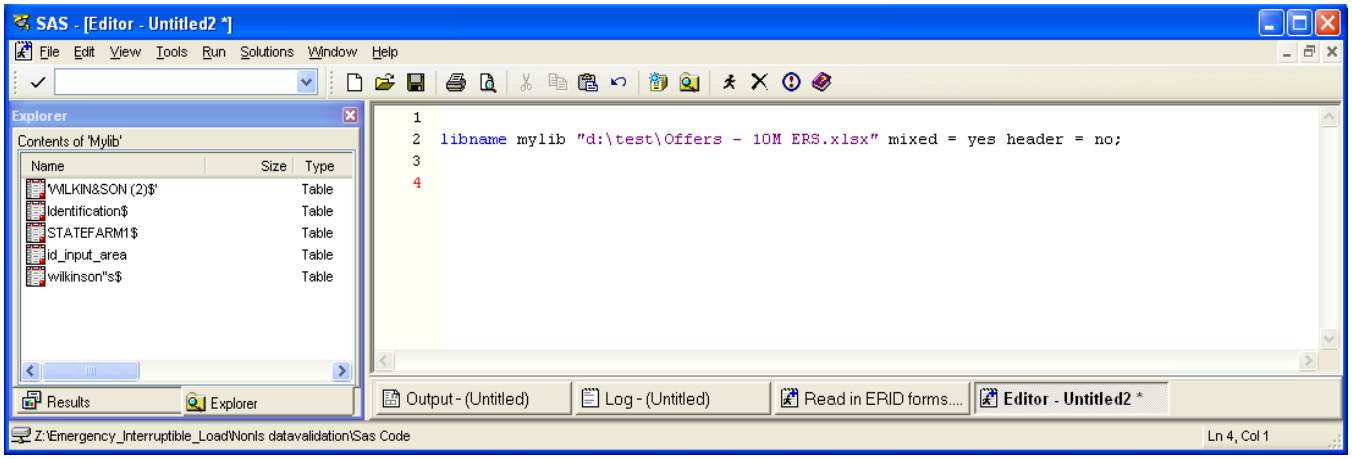
The complete code ERCOT uses to read these workbooks is included in Appendix A. The remainder of this section will discuss what the code is doing step by step.

The first section of code, shown below, sets up several macro variables including version to compare against the ones on the submitted form to make sure the most recent version is being used. All forms to be processed are dropped into the folder specified by the '&offerpath' macro variable. The first data step creates 'fnames_list' by scanning the folder and selecting all Excel files found in the folder. The second data step creates a macro variable, '&numfiles', which contains the number of files to be processed.

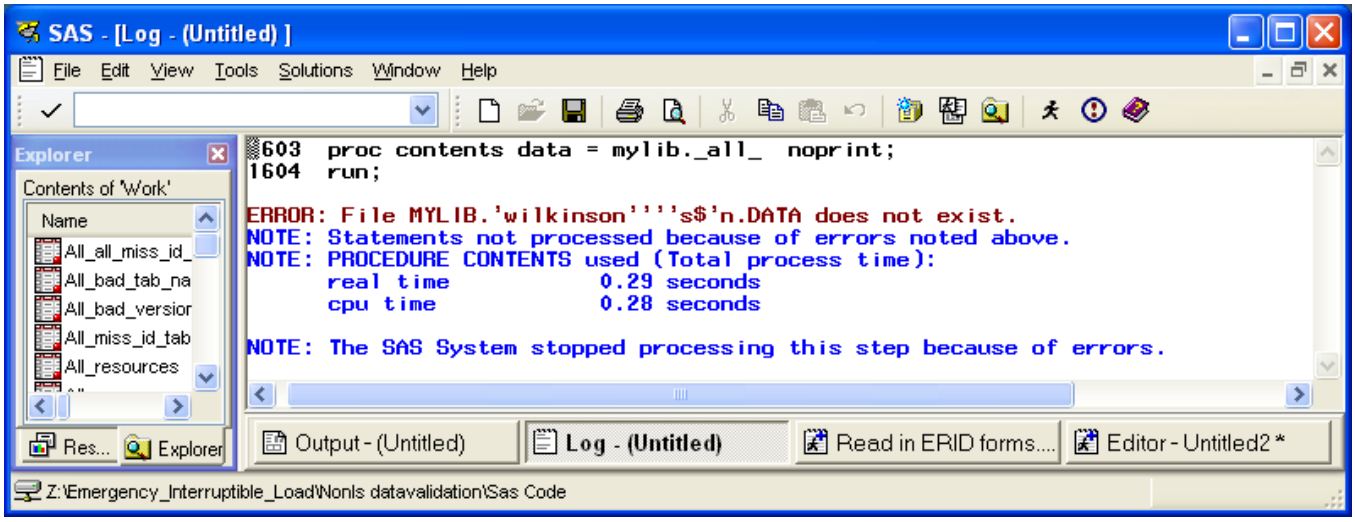


Most of the rest of the code is nested inside a macro called 'read_work_book'. The major macro do-loop inside this macro cycles through the workbooks one at a time using the '&numfiles' macro variable created in the preceding step.

The libname statement shown below is an example of mapping to an Excel workbook using the Excel libname engine. The SAS explorer window for 'mylib' shows the tab names and named ranges for the particular Excel workbook … the tab names end with dollar signs and named ranges do not. The tab name, wilkinson's, in the workbook shows up in the explorer window as wilkinson''s; double clicking on the table in the explorer window, however, generates a message that the table either does not exist or cannot be displayed.



Below is a sample of the normal log output obtained when running the proc contents on the workbook; this is an example of the kind of errors issued when tab names containing special characters are processed.



The section of code shown below shows another way to generate a list of tab names by using proc datasets inside an ODS group to create a SAS dataset, 'tabnames', of the tabs and named ranges in the workbook. This method is the most successful I've found for producing a complete list of names while avoiding errors for tab names that contain special characters (', &, (,), space). A viewtable of 'tabnames' for a test workbook with many combinations of potential problem names is shown following the section of SAS code. In general, if SAS encounters a tab name with any of these

special characters other than a single quote, single quotes are added to the front and back of the tab name. If SAS encounters a single quote as part of the tab name, it turns the single quote into two single quotes.

The subsequent data step creates two datasets, 'tab_names_1' and 'bad_tab_names'. Tab names written to 'tab_names_1' that had a single quote within the tab name (that SAS doubled) are translated back to having a single quote, and single quotes added at the beginning and end of the tab name. Tab names that had a single quote and an '&' within the tab name are written to 'bad_tab_names'; I have not found a way to manipulate these kind of tab names to get SAS to read them. Further down in the code the 'bad_tab_names' data set is accumulated into the 'all_ bad_tab_names' to make available a list of all Excel files with problem tab names. For the time being, the listed workbooks are modified manually by ERCOT to eliminate the special characters and reprocessed. If the 'bad_tab_names' dataset is not empty, an error message is displayed in the log, and no other tabs in the workbook are processed.

The data step also excludes several tab names with restricted use for this application: the 'Availability', 'Baseline', 'Exceptions', 'Identification', etc tabs either are reserved for subsequent use by ERCOT or are simply excluded. If the submitter has set up a print range in the workbook, that too is excluded from the 'tab_names_1' data set.

SAS - [Read in ERID forms.sas]

File Edit View Tools Run Solutions Window Help

```
44 %macro read_work_book;
45
46   %let first = 1;
47
48   %do i = 1 %to &numfiles.;
49
50     data _null_ ;
51
52       obsnum = input("&i.",4.);
53       set fnames_List point = obsnum;
54       call symputx("filename",put(filename,$300.));
55       stop;
56
57     run;
58
59     libname mylib "&offerpath.\&filename." mixed = yes header = no;
60
61     ods trace on;
62     ods output "Library Members" = tabnames;
63
64     proc datasets library = mylib;
65     quit;
66
67     ods output close;
68     ods listing;
```

```
70     %let numbad = 0;
71     data tab_names_1 (keep = filename tab_name)    bad_tab_names (keep = filename tab_name);
72
73       length tab_name $ 35 filename $ 300;
74       set tabnames;
75
76       filename = "&filename.";
77       if index(name, "'") > 0 then tab_name =  "'" || trim(tranwrd(name, "'", "'")) || "'";
78       else tab_name = name;
79
80       if length(trim(tab_name)) >= 4 then do;
81         if index(substr(tab_name, 2, length(trim(tab_name)) - 2), "'") > 0 &
82           index(substr(tab_name, 2, length(trim(tab_name)) - 2), "&") > 0 then do;
83           output bad_tab_names;
84           call symputx("numbad",1);
85         end;
86         else do;
87           if upcase(substr(tab_name,1,4))
88             in('AVAI', 'BASE', 'EXCE', 'ID_I', 'IDEN', 'R$', 'SHEE')
89             | index(tab_name,'Print_Area') > 0 then delete;
90           else output tab_names_1;
91         end;
92       end;
93       else if upcase(tab_name) ^= 'R$' then output tab_names_1;
94
95     run;
96
97     %if &numbad. = 1 %then %do;
98       %put ;
99       %put %str(ERROR: !!!!!!!!!!!!! Tab Name with Invalid Characters in Workbook !!!!!!!!!!!!!!!!!!  );
100      %put ;
101    %end;
```

Output - (Untitled) | Log - (Untitled) | Read in ERID forms... | Results Viewer - file:/...

Autosave complete | D:\test | Ln 85, Col 13

SAS - [VIEWTABLE: Work.Tabnames (Library Members)]

File Edit View Tools Data Solutions Window Help

	Variable Number	Name	Member Type	DBMS Member Type
1	1	'STATE FARM1$'	DATA	TABLE
2	2	'WILKIN&SON"s$'	DATA	TABLE
3	3	'WILKINSON(2)$'	DATA	TABLE
4	4	'a b c$'	DATA	TABLE
5	5	'a&b(2) c$'	DATA	TABLE
6	6	'a"b&c d$'	DATA	TABLE
7	7	'a"b&c(2)$'	DATA	TABLE
8	8	'a"sb&c(2) d$'	DATA	TABLE
9	9	'a(1) b$'	DATA	TABLE
10	10	'a(1)b(2)$'	DATA	TABLE
11	11	'state&farm$'	DATA	TABLE
12	12	'wilk &son$'	DATA	TABLE
13	13	'wilk&so&n$'	DATA	TABLE
14	14	'wilk&son(2)$'	DATA	TABLE
15	15	'wilkin son"s$'	DATA	TABLE
16	16	'wilkinson"s(2)$'	DATA	TABLE
17	17	Identification$	DATA	TABLE
18	18	a"sb"s$	DATA	TABLE
19	19	id_input_area	DATA	TABLE
20	20	wilkinson"s$	DATA	TABLE

Output - (Untitl... | Log - (Untitled) | Read in ERID... | Results View... | VIEWTABLE...

D:\test

SAS - [VIEWTABLE: Work.Tab_names_1]

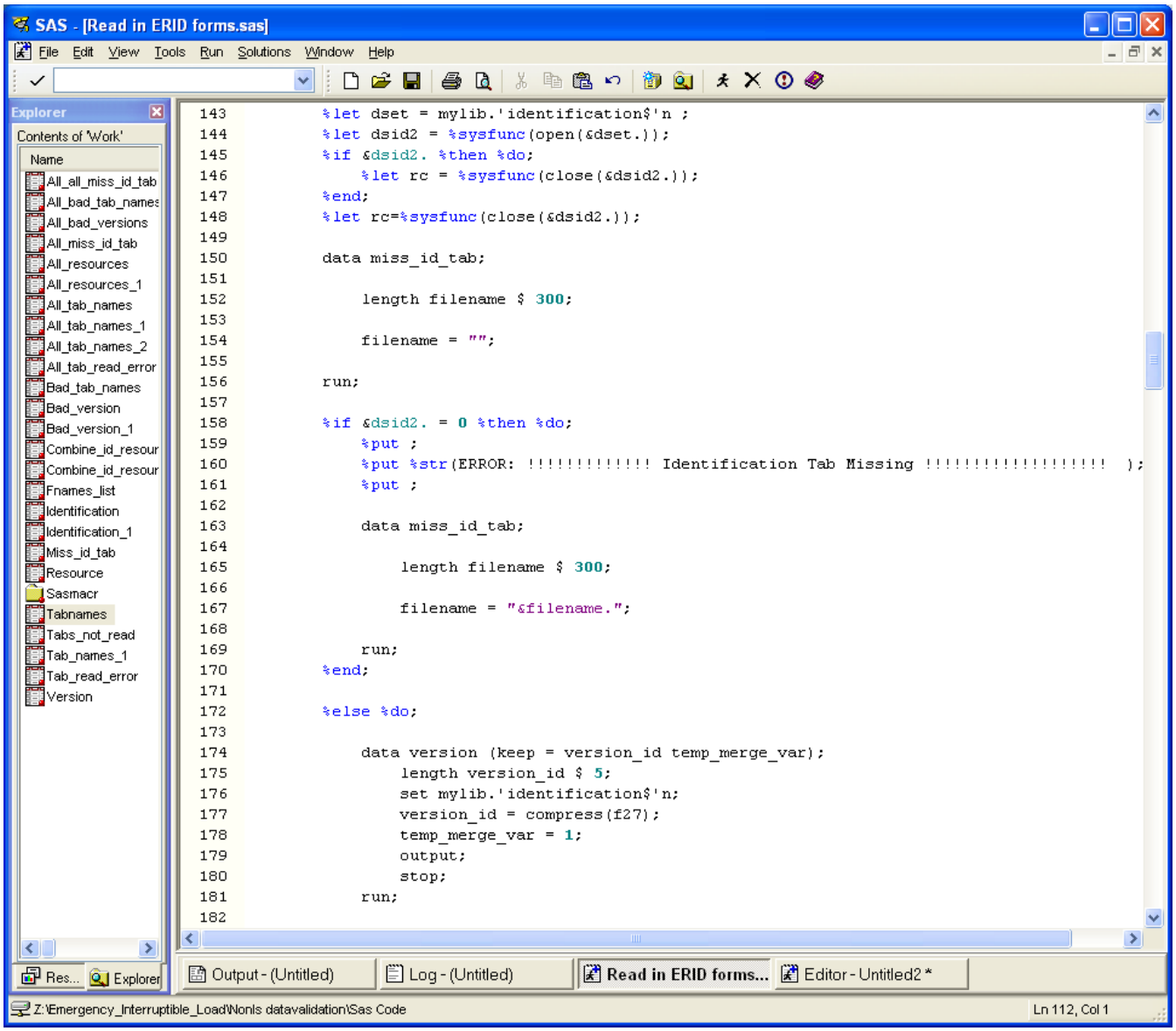
	tab_name	filename
1	'STATE FARM1$'	Offers - 10M ERS.xlsx
2	'WILKINSON(2)$'	Offers - 10M ERS.xlsx
3	'a b c$'	Offers - 10M ERS.xlsx
4	'a&b(2) c$'	Offers - 10M ERS.xlsx
5	'a(1) b$'	Offers - 10M ERS.xlsx
6	'a(1)b(2)$'	Offers - 10M ERS.xlsx
7	'state&farm$'	Offers - 10M ERS.xlsx
8	'wilk &son$'	Offers - 10M ERS.xlsx
9	'wilk&so&n$'	Offers - 10M ERS.xlsx
10	'wilk&son(2)$'	Offers - 10M ERS.xlsx
11	"wilkin son's$"	Offers - 10M ERS.xlsx
12	"wilkinson's(2)$"	Offers - 10M ERS.xlsx
13	'a'sb's$'	Offers - 10M ERS.xlsx
14	'wilkinson's$'	Offers - 10M ERS.xlsx

SAS - [VIEWTABLE: Work.Bad_tab_names]

	tab_name	filename
1	"WILKIN&SON's$"	Offers - 10M ERS.xlsx
2	"a'b&c d$"	Offers - 10M ERS.xlsx
3	"a'b&c(2)$"	Offers - 10M ERS.xlsx
4	"a'sb&c(2) d$"	Offers - 10M ERS.xlsx

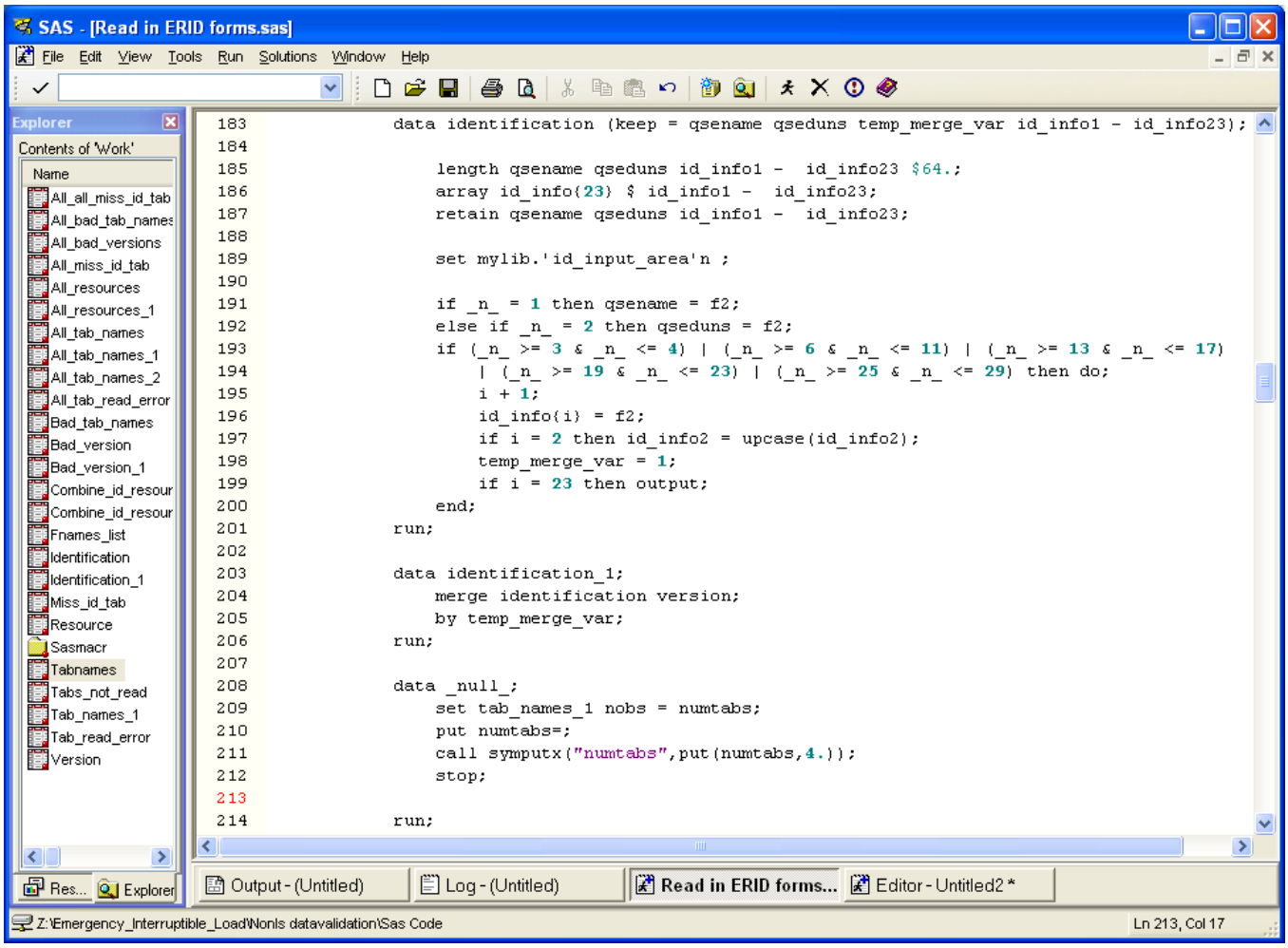
NOTE: You cannot shrink the window beyond this point.

The next section of code, shown below, checks whether the required 'Identification' tab is present in the workbook. If it is not present, an error message is displayed in the log, and no other tabs in the workbook are processed. The filenames of workbooks with a missing identification tab are output to the 'miss_id_tab' dataset; further down in the code the 'miss_id_tab' data set is accumulated into the 'all_ miss_id_tab' to make available a list of all Excel files with missing 'Identification' tabs. If the 'Identification' tab is present, the 'version' data set is created; this data step reads just the first row of the 'Identification' tab and keeps the version number stored in cell AA1.



The next section of code inputs identification information from the 'id_input_area' named data range; the range is two columns wide, and the data needed is in the second column of some of the rows in the range (the other rows are cosmetic merged cells to make the sections of the form stand out clearly). The QSE's name is entered on the first row of the range, and the QSE's Duns number is entered on the second row of the range. The remainder of the range contains contact information for the QSE and is read into the 'id_info' array. Note that because the headers option was set to no on the libname statement, the incoming variable of interest is always 'f2'. Since each row of the workbook is treated as an observation by SAS, all needed variables are retained until the last row of the range has been read at which point the observation is output to the 'identification' dataset.

The 'identification_1' dataset is the merge of the identification information with the version number, and the final data step in this section of code creates a macro variable, '&numtabs', for the number of tabs in the workbook. This macro variable is used in the macro do loop to sequence through the tabs found in the workbook.



The next section of code shows the beginning of the macro do loop; the first data step does a direct access read of the 'tab_names_1' data set, based on the iteration of the macro do loop, to create a macro variable, '&sheet', containing the 'sheet' statement to be used with the subsequent proc import step. If the modified tab name is enclosed in single quotes and also contains a single quote, the 'sheet' statement is created with double quotes to allow the proc import to read that particular tab. Proc import is the only way I've found to read these tabs. Note again that the proc import uses the following options: 'mixed = yes getnames = no and scantext = yes'. The 'mixed = yes' option tells SAS that all cells are to be treated as characters … all conversions to numeric variables are handled in the subsequent data step. The 'getnames = no' option tells SAS that data begins on the first row of the workbook; otherwise SAS would interpret the first row as containing variable names. The 'getnames = no' option causes SAS to use automatic variable names: column A becomes F1, column B becomes F2, etc. The 'scantext = yes' option tells SAS to scan each column to determine the maximum length to use for each character variable.

SAS - [Read in ERID forms.sas]

```
%do j = 1 %to &numtabs.;

  data tab_names_2 (keep = tab_name temp_merge_var) ;

    length tab_name_quote $ 300;

    obsnum = input("&j.",4.);
    set tab_names_1 point = obsnum;

    if substr(tab_name, 1, 1) = "'" & substr(tab_name, length(trim(tab_name)), 1) = "'"
      & index(substr(tab_name, 2, length(trim(tab_name)) - 2), "'") > 0 then do;
      tab_name_quote = 'sheet = "' || trim(put(tab_name,$300.)) || '"';
      call symputx("sheet", tab_name_quote);
    end;

    else if substr(tab_name, 1, 1) = "'" & substr(tab_name, length(trim(tab_name)), 1) = "'" then do;
      tab_name_quote = "sheet = " || trim(put(tab_name,$300.));
      call symputx("sheet", tab_name_quote);
    end;

    else do;
      tab_name_quote = "sheet = '" || trim(put(tab_name,$300.)) || "'";
      call symputx("sheet", tab_name_quote);
    end;

    temp_merge_var = 1;
    output;
    stop;

  run;


  proc import datafile="&eridpath.\&filename." replace out = import_resource dbms = excel;
    mixed = yes;
    getnames = no;
    scantext = yes;
    &sheet.;
  run;

  data import_resource_1;

      set import_resource;

      temp_merge_var = 1;

  run;
```

Output - (Untitled) | Log - (Untitled) | Read in ERID forms... | Results Viewer - SA...

File saved successfully. D:\test Ln 203, Col 43

The next data step parses and converts the data imported from each tab in the workbook and outputs it to the 'resource' data set; the code is set up to pull data from the f-variable associated with the appropriate cell locations based on whether the tab being input is an 'Alt' tab or a standard resource tab. The _n_ automatic SAS variable in combination with the f1 – f27 variables are used as positions within the row to define each of the variables being read from the workbook … cells with labeling information and cells which are intended to be blank for aesthetic purposes are skipped over. As mentioned above, because of the 'mixed = yes' option, all variables were read in as characters. In this step, they are assigned to variables kept in the output dataset with the lengths of character variables controlled specifically by the code for each variable and with numeric variables, if any, converted using the input function. As with the 'Identification' tab, since each row of the workbook is treated as an input observation by SAS, all needed variables are retained until the last row needed for the observation is output to the 'resource' dataset. Observations are output for each incoming row with customer data provided that the esiid or unique_meter_id variables are greater than blank.

As with the ‘Identification’ tab, a version number for each tab is read from cell AA1 and used for subsequent validation to identify whether the QSE is using the current workbook version.

SAS - [Read in ERID forms.sas]

File Edit View Tools Run Solutions Window Help

```
228         data resource (keep = version_resource filename tab_name renewal_opt_in resource_type
229             resource_name baseline analyze_offer1 - analyze_offer4
230             esiid sub_metering unique_meter_id load_resource lr_in_as site_ownr site_name
231             street city zipcode tspdsp_ia tspdsp_ia_year tspdsp_name tspdsp_duns
232             noie_area wsale_meter priv_net onsite_gen gen_type load_desc
233             temp_merge_var eridnum rundate);
234
235           length version_resource $ 5 filename $ 300 tab_name $ 35
236             resource_type resource_name  baseline $ 64 renewal_opt_in $ 3
237             esiid $ 64 sub_metering $ 3 unique_meter_id $ 64 load_resource lr_in_as $ 3
238             site_ownr site_name $ 64 street city $64 zipcode $5. tspdsp_ia $ 3
239             tspdsp_ia_year  tspdsp_name tspdsp_duns $64. noie_area wsale_meter
240             priv_net onsite_gen $3. gen_type $30. load_desc $256.
241             analyze_offer1 - analyze_offer4 $ 3;
242
243           format rundate datetime20.;
244
245           retain version_resource temp_merge_var filename tab_name renewal_opt_in resource_type
246             resource_name baseline analyze_offer1 - analyze_offer4 eridnum rundate;
247
248           merge import_resource_1 tab_names_2;
249           by temp_merge_var;
250
251               if tab_name = 'Alt' then baseline = 'ALTERNATE';
252               if _n_ = 1 then do;
253                 temp_merge_var = 1;
254                 *tab_name = substr("&sheet.", 9, 1);
255                 tab_name = translate(tab_name, " ", "$");
256                 filename = "&filename.";
257                 rundate = datetime();
258                 eridnum = round(100 * (rundate - "01jun2009:01:00:00"dt),1);
259                 version_resource = f27;
260               end;
261               else if _n_ >= 2 & _n_ <= 5 & tab_name ^= 'Alt' then do;
262                 if _n_ = 2 then do;
263                   renewal_opt_in = '';
264                   resource_type = f3;
265                   analyze_offer1 = upcase(f5);
266                 end;
267                 else if _n_ = 3 then analyze_offer2 = upcase(f5);
268                 else if _n_ = 4 then do;
269                   resource_name = f1;
270                   if resource_name = '' then resource_name = 'Missing Resource Name';
271                   baseline = upcase(f3);
272                   analyze_offer3 = upcase(f5);
273                 end;
274                 else if _n_ = 5 then analyze_offer4 = upcase(f5);
275               end;
276               else if (tab_name = 'Alt' & _n_ >= 4) | (tab_name ^= 'Alt' & _n_ >= 7) then do;
```

```
                esiid = compress(f1, "' ");
                sub_metering = upcase(f2);
                unique_meter_id = f3;
                if esiid > '' & unique_meter_id = '' & sub_metering = ''
                  then sub_metering = 'NO';
                load_resource = upcase(f4);
                lr_in_as = upcase(f5);
                site_ownr = f6;
                site_name = f7;
                street = f8;
                city = f9;
                zipcode = f10;
                tspdsp_ia = upcase(f11);
                tspdsp_ia_year = f12;
                tspdsp_name = f13;
                tspdsp_duns = f14;
                noie_area = upcase(f15);
                wsale_meter = upcase(f16);
                priv_net = upcase(f17);
                onsite_gen = upcase(f18);
                gen_type = f19;
                load_desc = f20;
                if esiid > '' | unique_meter_id > '' then output;
                else if esiid = '' & unique_meter_id = ''
                  & (sub_metering > '' | unique_meter_id > '' | load_resource > '' | lr_in_as > ''
                  | site_ownr > '' | site_name > '' | street > '' | city > '' | zipcode > ''
                  | tspdsp_ia > '' | tspdsp_ia_year > '' | tspdsp_name > '' | tspdsp_duns > ''
                  | noie_area > '' | wsale_meter > '' | priv_net > '' | onsite_gen > ''
                  | gen_type > '' | load_desc > '') then
                  put /// 'esiid and umi both blank ' filename = tab_name = /// '';
              end;
            run;
```

Output - (Untitled) | Log - (Untitled) | Read in ERID forms... | Results Viewer - SA...

File saved successfully. D:\test Ln 221, Col 1

The next section of code determines the number of observations in ‘resource’ dataset and if the number is greater than zero merges the identification and resource information. The ‘combine_id_resource_1’ dataset contains observations found with valid version numbers, and the ‘bad_version_1’ dataset identifies resources with invalid version numbers.

SAS - [Read in ERID forms.sas]

```
307                 %let dset = work.resource;
308                 %let dsid3 = %sysfunc(open(&dset));
309
310                 %if &dsid3 %then %do;
311
312                     %let numloads =%sysfunc(attrn(&dsid3,NOBS));
313                     %let rc = %sysfunc(close(&dsid3));
314
315                 %end;
316
317                 %let rc=%sysfunc(close(&dsid3));
318                 %if &numloads. > 0 %then %do;
319
320                     data combine_id_resource bad_version (keep = qsename eridnum sub_eridnum);
321
322                         merge identification_1 (in = in1) resource (in = in2);
323                         by temp_merge_var;
324
325                         if  ^in1 | ^in2 | compress(version_id) ^= "&version."
326                             | compress(version_resource) ^= "&version." then do;
327                             output bad_version;
328                             call symputx("version_error", "Yes");
329                         end;
330                         output combine_id_resource;
331                     run;
332
333                     proc sort data = bad_version nodupkey;
334                         by eridnum sub_eridnum;
335                     run;
336
337                     proc sort data = combine_id_resource;
338                         by eridnum sub_eridnum;
339                     run;
340
341                     data combine_id_resource_1 bad_version_1;
342                         merge combine_id_resource bad_version (in = in1);
343                         by eridnum sub_eridnum;
344                         if ^in1 then output combine_id_resource_1;
345                         else output bad_version_1;
346                     run;
```

Output - (Untitled) | Log - (Untitled) | Read in ERID forms... | Editor - Untitled2 *

Z:\Emergency_Interruptible_Load\NonIs datavalidation\Sas Code Ln 340, Col 1

The final piece of code in the 'read_work_book' macro accumulates all the resource and error data sets across all workbooks and tabs within workbooks. Finally, the code clears the libname assigned to the Excel workbook. An additional section of code in a macro called 'check_errors' is shown in the Appendix; this code is included to display error messages at the end of the log. The messages trigger investigations into problems with any of the workbooks processed; in many cases the volume of workbooks and tabs produced is very large causing the log to be very long and earlier error messages generated to be easily overlooked.

SAS - [Read in ERID forms.sas]

File Edit View Tools Run Solutions Window Help

```
351          %if &first. = 1 %then %do;
352
353            data all_resources;
354              set combine_id_resource_1;
355            run;
356
357            data all_bad_versions;
358              set bad_version_1;
359            run;
360
361            data all_tab_names;
362              set tab_names_1;
363            run;
364
365            data all_bad_tab_names;
366              set bad_tab_names;
367            run;
368
369            data all_miss_id_tab;
370              set miss_id_tab (where = (filename > ''));
371            run;
372
373
374            %let first = 0;
375
376          %end;
377          %else %do;
378
379            proc append base = all_resources data = combine_id_resource_1;
380            run;
381
382            proc append base = all_bad_versions data = bad_version_1;
383            run;
384
385            proc append base = all_tab_names data = tab_names_1;
386            run;
387
388
389          %end;
390        %end;
391      %end;
392    %end;
```

```
393
394     proc append base = all_bad_tab_names data = bad_tab_names;
395     run;
396
397     proc append base = all_miss_id_tab data = miss_id_tab (where = (filename > '')); 
398     run;
399
400   %end;
401
402 %mend;
403
404 %read_work_book;
405
406 libname mylib clear;
```

Output - (Untitled) | Log - (Untitled) | Read in ERID forms... | Results Viewer - SA...

D:\test | Ln 350, Col 1

WRITING OUT PROCESSED FORMS

Resources that are read and output with the code described in the previous section proceed to an extensive validation step, and if successful, continue with historical usage data analysis. SAS datasets summarizing the validation and analysis are produced and stored for later use when approved forms are created to be returned to the QSEs. These forms are then completed by QSEs and returned to ERCOT in the form of an offer to provide the service.

The complete code for writing out workbooks is contained in Appendix B. The process of writing out the approved workbooks consists of using the Excel libname engine to move data from SAS datasets to Excel and using DDE commands to open, save and close workbooks, and to execute various user defined Excel macros.

To get started, the options command sets the noxwait and noxsync options; this is needed to get the DDE commands to function properly. Some macro variables are set up to establish paths to the permanent SAS datasets produced by the validation and analysis steps and a path to the Excel templates used to generate the workbooks. The data step to create 'okay_ers_inputs' is fairly straight forward; code to take note of is the creation of the 'blout1 – blout4' variables … these are used later on to populate a drop-down list in the workbook to provide baseline options for the specific resource. When the QSE returns the form, one of these allowed options must be selected.

SAS - [Step 15 DDE_Code_Write out ok erids.sas]

```
options noxwait noxsync symbolgen mlogic mprint linesize = 200 msglevel = i;


%let contract_period = 2012 OctJan13;
%let local_drive = d:;                    * enter the drive letter for the templates and output path ;

/* path to write xls files and to see if they already exist for this batch */
%let eridpath = Emergency_Interruptible_Load\&contract_period.\00 resource Identification\3 Outgoing erids\okay;
%let templatepath = Emergency_Interruptible_Load\&contract_period.\00 resource Identification\DDE Templates;

libname perm "Z:\Emergency_Interruptible_Load\&contract_period.\00 resource Identification\sas datasets";

    /*****************************************************/
    /* create dataset of qsenames and count of erids  */
    /*****************************************************/
data okay_ers_inputs (keep = eridnum qsename qseduns renewal_opt_in resource_name resource_type
        load_type baseline analyze_offer1 - analyze_offer4  blout1 - blout4
        load ss_gen non_ss_gen esiid sub_metering unique_meter_id load_resource
        lr_in_as site_ownr site_name site_name street city zipcode tspdsp_ia tspdsp_ia_year
        tspdsp_name tspdsp_duns noie_area wsale_meter priv_net onsite_gen gen_type load_desc
        id_info1 - id_info23);

    length eridnum 8 qsename qseduns $ 64 renewal_opt_in $ 3 resource_type resource_name baseline $ 64
        analyze_offer1 - analyze_offer4 $ 3 blout1 - blout4 blout1 - blout4 load ss_gen non_ss_gen $ 30
        esiid $ 64 sub_metering $ 3 unique_meter_id $ 64 load_resource lr_in_as $ 3 site_ownr site_name
        street city $ 64 zipcode $ 5 tspdsp_ia $ 3 tspdsp_ia_year tspdsp_name tspdsp_duns $ 64
        noie_area wsale_meter priv_net onsite_gen $ 3 gen_type $ 30 load_desc $ 256
        id_info1 - id_info23 $ 64;

    set perm.ers_inputs_okay_final;

    renewal_opt_in = '';
    load = 'Load';
    ss_gen = 'Self-Serving Gen';
    non_ss_gen= 'Non Self-Serving Gen';
    if bline1 = 1 then blout1 = 'DEFAULT-REGRESSION';
    if bline2 = 2 then blout2 = 'DEFAULT-MID 8 OF 10';
    if bline3 = 3 then blout3 = 'DEFAULT-MATCHING DAY PAIR';
    blout4 = 'ALTERNATE';
    if baseline = '' & bline1 = . & bline2 = . & bline3 = . then baseline = 'ALTERNATE';

run;
```

Output - (Untitled) | Log - (Untitled) | Step 15 DDE_Code...

File saved successfully. Z:\Emergency_Interruptible_Load\2012 Ln 16, Col 57

The next data step begins the process of creating tab names for the workbook to be created using the QSE provided resource name as a basis. Some clean-up is done to remove any special characters that either will not be accepted by Excel as tab names or that would create problems when SAS subsequently reads the workbook back in when the submitted by the QSE as an offer. An important statement included in this part of the code is the final modification of the tab name using the upcase function; the statement is needed because, unlike SAS, Excel will treat tab names with the same letters and different cases as the same tab. By using the upcase function, SAS can recognize in the subsequent step that a duplicate tab has been generated, and modify tab names to ensure separate tabs are written.

SAS - [Step 15 DDE_Code_Write out ok erids.sas]

```
45 data okay_ers_inputs_1 (drop = temp_tab_name more i numbers letters);
46
47     length tab_name $ 29 temp_tab_name $ 64;
48
49     set okay_ers_inputs;
50
51     if resource_name = '' then tab_name = 'Alt';
52
53     else do;
54         temp_tab_name = compress(resource_name, "'");
55         temp_tab_name = compress(temp_tab_name, '"');
56         temp_tab_name = translate(temp_tab_name,"                                    ",
57                                                  " !@#$%^&*()_+-={}|[]\:;<>?,./;");
58         temp_tab_name = compbl(temp_tab_name);
59         temp_tab_name = translate(temp_tab_name,"~", " ");
60         more = 1;
61         do i = 64 to 1 by - 1 while (more = 1);
62             if substr(temp_tab_name,i,1) = '~' then substr(temp_tab_name,i,1) = ' ';
63             else more = 0;
64         end;
65         tab_name = compbl(substr(translate(temp_tab_name, '_', '~'),1,26));
66         if substr(tab_name, 26,1) = '_' then tab_name = substr(tab_name, 1, 25);
67         letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
68         numbers = '1234567890';
69         if verify(substr(tab_name,1,3), letters) = 0 & verify(compress(substr(tab_name,4)), numbers) = 0
70             & substr(tab_name,4) > '' then tab_name = substr(tab_name,1,3) || '_' || substr(tab_name,4);
71         else if verify(substr(tab_name,1,2), letters) = 0 & verify(compress(substr(tab_name,3)), numbers) = 0
72             & substr(tab_name,3) > '' then tab_name = substr(tab_name,1,2) || '_' || substr(tab_name,3);
73         else if verify(substr(tab_name,1,1), letters) = 0 & verify(compress(substr(tab_name,2)), numbers) = 0
74             & substr(tab_name,2) > '' then tab_name = substr(tab_name,1,1) || '_' || substr(tab_name,2);
75         if verify(substr(tab_name,1,1), letters) ^= 0 then tab_name = '_' || tab_name;
76         tab_name = upcase(tab_name);
77     end;
78
79 run;
80
81 proc sort data = okay_ers_inputs_1;
82     by qseduns tab_name eridnum;
83 run;
```

Output - (Untitled) | Log - (Untitled) | Step 15 DDE_Code...

Autosave complete | Z:\Emergency_Interruptible_Load\2012 | Ln 52, Col 5

The next section of code splits the input data into two datasets, one containing site information to be written to an Alt tab (because the QSE submitted it on an Alt tab), and one containing resource level information for sites submitted on a standard resource tab. The code then deals with the possibility of duplicate tab names … submission of duplicate resource names is allowed and the manipulation performed in the preceding step can also result in the duplication of tab names. The code below appends an index number to any duplicate tab names found on a QSE-by-QSE basis to eliminate the duplication … including any duplication created in the previous step by the upcase function. The

remaining steps in this section of code determine how many QSEs have resources in the batch of workbooks being processed; a single workbook of resources is produced for each QSE in the batch. The macro variable '&numqses' is used in the macro do loop to sequence through all QSEs in the batch, and the 'qse_list' dataset contains the list of QSE names and Duns numbers included in the batch. The 'qse_erid_list' contains a list of all resource tabs that need to be created across all the workbooks in the batch.

```
85  data okay_ers_rtab_inputs okay_ers_alt_inputs ;
86
87      drop count;
88
89      set okay_ers_inputs_1;
90      by qseduns tab_name eridnum;
91
92      if tab_name ^= 'Alt' then do;
93          if first.tab_name then count = 0;
94          if first.eridnum then count + 1;
95          if count > 1 & count < 100 then
96              tab_name = translate(compbl(substr(tab_name,1,26) || put(count,2.)),'_',' ');
97          output okay_ers_rtab_inputs;
98      end;
99      else output okay_ers_alt_inputs;
100
101 run;
102
103 proc sort data = okay_ers_rtab_inputs out = qse_erid_list nodupkey;
104     by qseduns tab_name eridnum;
105 run;
106
107 proc freq data = okay_ers_inputs_1 noprint;
108     tables qsename * qseduns  / out = qse_list;
109 run;
110
111     /*********************************************/
112     /* create macro variable of number of qses  */
113     /*********************************************/
114
115 %let numqses = 0;
116
117 data _null_;
118     set qse_list nobs = numqses;
119     call symputx("numqses",put(numqses,5.));
120     stop;
121 run;
```

The remainder of the code is contained in the 'run_qse' macro, which cycles through all QSEs in the batch and produces a workbook for each QSE. The 'fill' dataset is just a placeholder dataset to be used later in creating tabs for each of the resources being output. The 'qse_erid_list_1' dataset is a subset of the entire list of QSE's tabs specific to the QSE being processed during the iteration of the macro do loop. The last part of this section of code determines which Excel

template should be used for the QSE being processed. If an 'Alt' tab is to be created, the 'ID_Alt_Template.xlsx' template, is chosen; otherwise, the 'ID_Template' is chosen. The macro variable, '&id_template', is set up to pass the template selection on to the subsequent code.

SAS - [Step 15 DDE_Code_Write out ok erids.sas]

```
123 %macro run_qse;
124
125     data fill;
126         a = 1; b = 1; c = 1;
127         if a > 1;
128     run;
129
130         /* loop through QSEs */
131     %if &numqses. > 0 %then %do i = 1 %to &numqses.;
132
133         data _null_;
134             obsnum = input("&i",4.);
135             set qse_list point = obsnum;
136             call symputx("qseduns",put(qseduns,$64.));
137             call symputx("qseduns_fname",put(input(qseduns,13.),z16.));
138             stop;
139         run;
140
141             /* set macro numerids to the number of r tabs for qse being processed */
142
143         %let numerids = 0;
144
145         /* create a file of eridnums for this qse */
146         data qse_erid_list_1 ;
147             set qse_erid_list (where = (qseduns = "&qseduns."));
148             numerids + 1;
149             call  symputx("numerids",numerids);
150         run;
151
152         /* set macro variable for the template to be used (with or without and Alt tab) */
153
154         %let ID_Template = ID_Template;
155
156         data _null_;
157
158             set okay_ers_alt_inputs (where = (qseduns = "&qseduns."));
159             call symputx("ID_Template", "ID_Alt_Template");
160             stop;
161         run;
```

The next data step issues system commands to create a copy of the appropriate template; this is done to prevent the original template from getting corrupted during the processing. The following macro do loop uses proc export to create tabs in the copied template workbook using the names in 'qse_erid_list_1' that was generated in the earlier step. The placeholder dataset, 'fill', is exported to each of the tabs created; this is done, because proc export does not run unless

it has a dataset to export. In earlier versions of this code, the tabs were created using DDE commands, and we've found that using proc export is a better alternative in terms of both speed and reliability.

SAS - [Step 15 DDE_Code_Write out ok erids.sas]

```
165         data _null_;
166             length command $ 200;
167             command="&local_drive.";
168             call system(command);
169             command="cd\&templatepath.\";
170             call system(command);
171             command="copy &ID_Template..xlsx &ID_Template._copy.xlsx";
172             call system(command);
173         run;
174
175     /* cycle through all tab_names for the qse being processed and proc export the tab to the */
176     /* template copy workbook     note: fill is an empty dataset to export */
177
178         %if &numerids. > 0 %then %do j = 1 %to &numerids.;
179
180
181         /* create tabs for all the R tab_names needed for this qse */
182             data _null_;
183                 obsnum = input("&j",4.);
184                 set qse_erid_list_1 point = obsnum;
185                 call symputx("tab_name", tab_name);
186                 stop;
187             run;
188
189             proc export data = fill outfile =
190                 "&local_drive.\&templatepath.\&ID_Template._copy.xlsx";
191                 sheet="&tab_name.";
192             run;
193
194         %end;
```

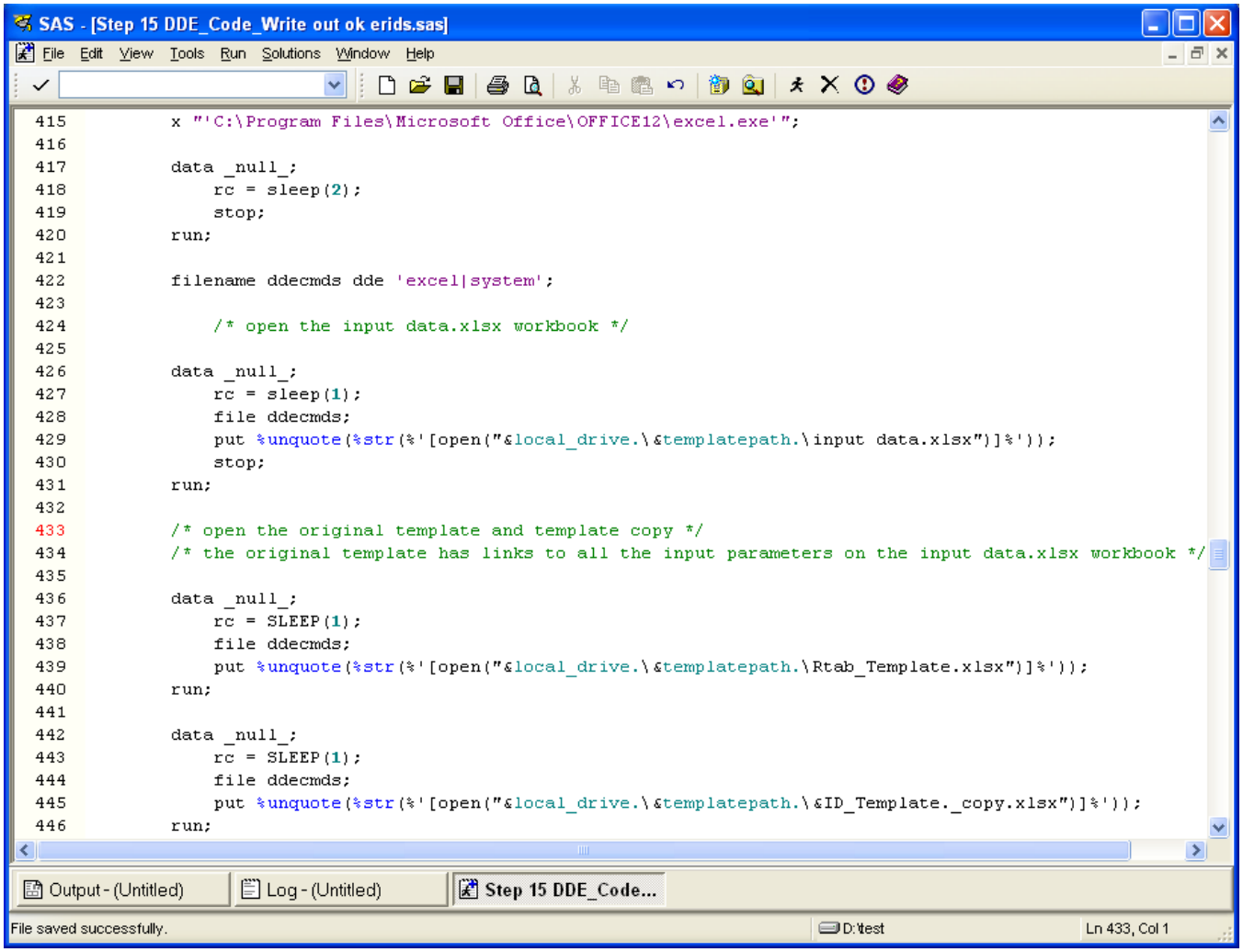
Output - (Untitled) | Log - (Untitled) | Step 15 DDE_Code...

Autosave complete | Z:\Emergency_Interruptible_Loac | Ln 167, Col 37

The next set of code, which is not shown below, formats three summary reports for the QSE being processed and exports them with proc export to the workbook being created for the QSE. After those exports are complete, the code below runs; the first line uses an X command to launch Excel, and must be coded with the path to the actual excel.exe on the pc running the code. The first data step causes SAS to pause and give Excel time to open and the filename statement allows SAS to send DDE commands to Excel.

The next three data steps send DDE commands to Excel to open the workbooks needed to send data from SAS to Excel. The open command is constructed in the required format using a macro variable for the path. Double quotes are used inside the 'open' command to allow the macro variables to be resolved, the single quotes are needed by the syntax of the DDE command, and the %unquote unmasks the single quotes created by the %str function to turn the single quotes into values the SAS compiler recognizes. The sleep commands are all used to give time for the Excel command to complete before the next command is issued by SAS.

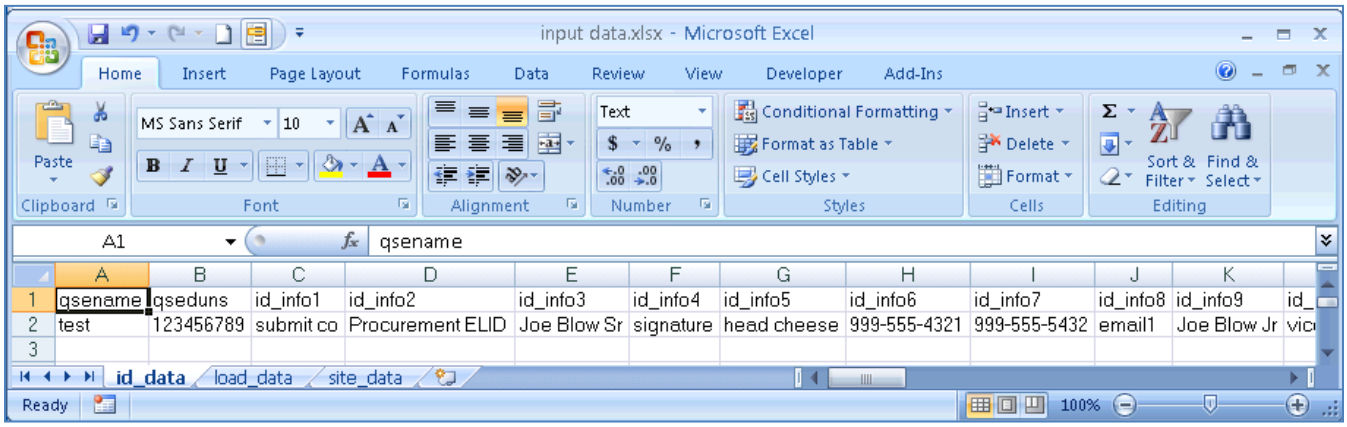
The first workbook opened is named 'input data.xlsx'; SAS will later write data to this workbook that is needed to populate tabs in the QSE workbook. The second workbook opened is named 'rtab_template.xlsx'; this workbook is the template used for creating resource tabs, and it contains links to 'input data.xlsx'. The third workbook is named either 'id_template_copy.xlsx' or 'id_alt_template_copy.xlsx' depending on whether the QSE workbook being created needs and 'Alt' tab or not. This workbook contains a tab named 'Identification' and is the template for that tab; the tab has links to 'input data.xlsx'. If the' id_alt_template_copy.xlsx' is used, it has a tab named 'Alt' and is the template for that tab; the tab also has links to 'input data.xlsx'.



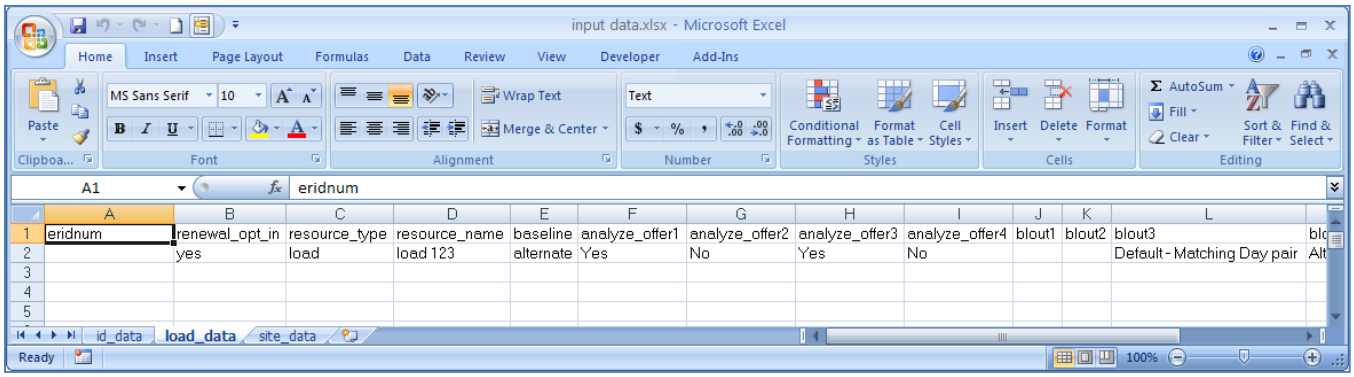
A portion of the input data.xlsx workbook is shown below; the workbook has three tabs: 'id_data', 'load_data' and 'site_data' that were set up for the three different types of data to be written to the workbook. SAS will export data needed for the' Identification' tab to the 'id_data' tab; resource level data to the 'load_data' tab; and site (customer) level data to the 'site_data' tab. The workbook is set up with a named range, 'id_inputs', which refers to cells A1 – AA2; this named range is used by SAS as the output location, when it writes to the 'id_data' tab.

Note that only one row is used for data going to the 'Identification' tab. Similarly, the workbook is set up with a named range, 'load_inputs', which refers to cells A1 – P2; this named range is used by SAS as the output location when it writes resource-level to the 'load_data' tab; only one row is used for resource-level data. Finally, the workbook has a named

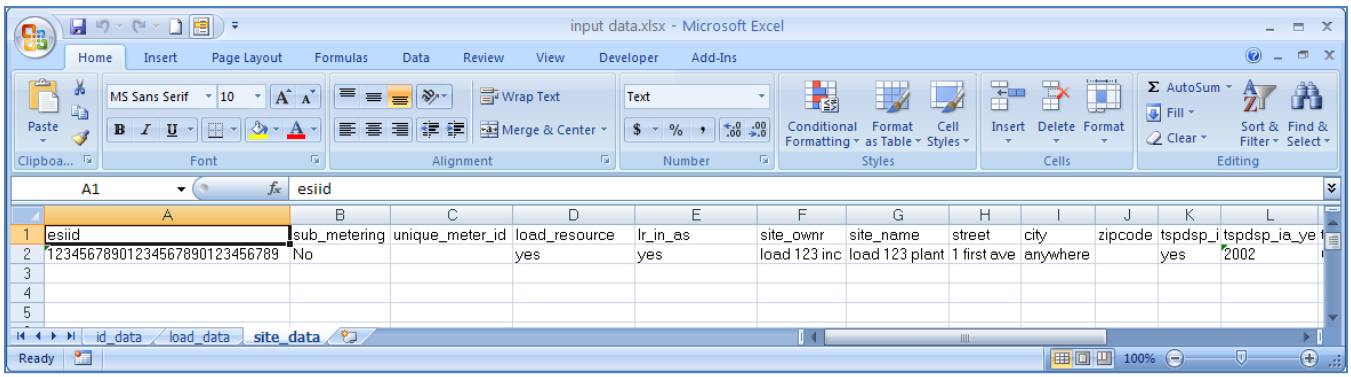
range, ‘site_inputs’, which refers to cells A1 – T1000, that is used by SAS as the output location, when it writes site-level data to the ‘site_data’ tab.



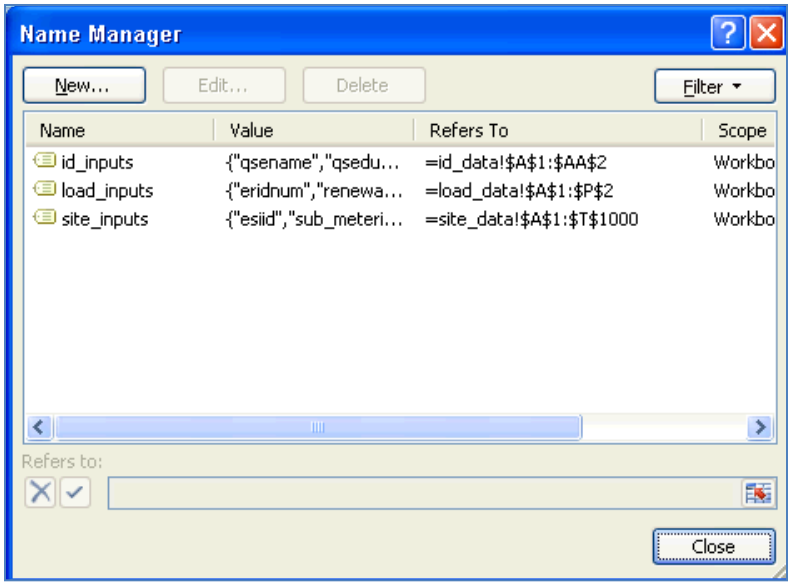
qsename	qseduns	id_info1	id_info2	id_info3	id_info4	id_info5	id_info6	id_info7	id_info8	id_info9	id_
test	123456789	submit co	Procurement ELID	Joe Blow Sr	signature	head cheese	999-555-4321	999-555-5432	email1	Joe Blow Jr	vic



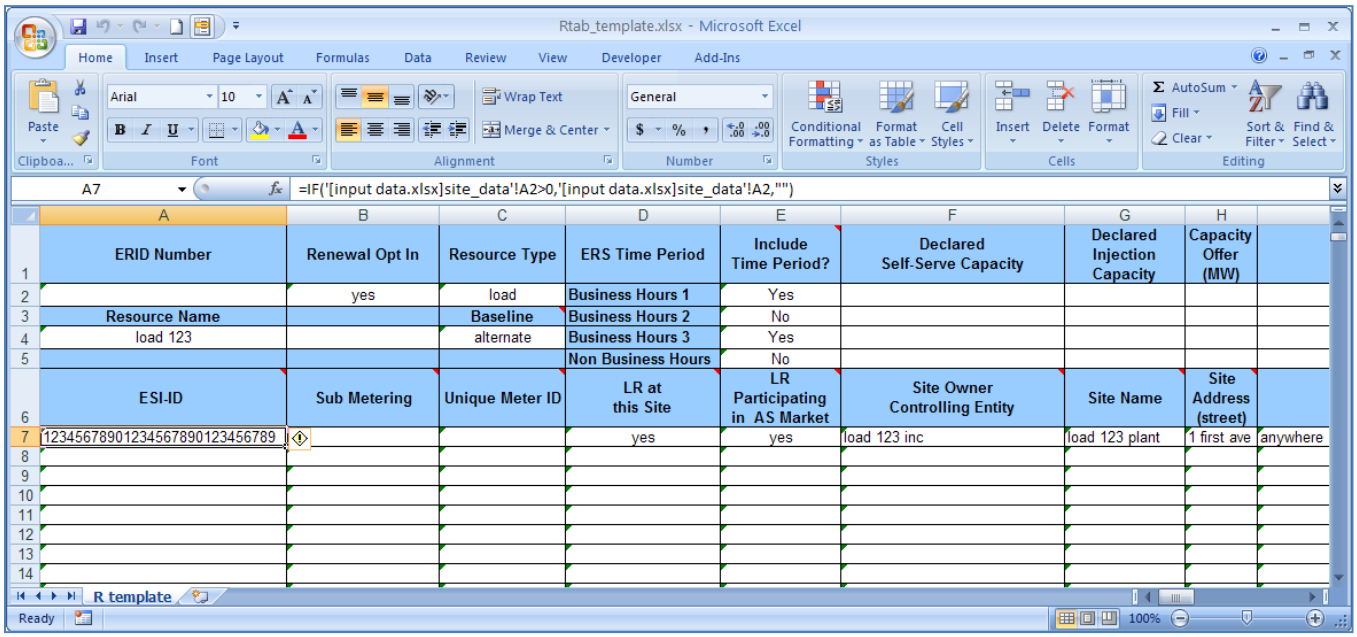
eridnum	renewal_opt_in	resource_type	resource_name	baseline	analyze_offer1	analyze_offer2	analyze_offer3	analyze_offer4	blout1	blout2	blout3	bl
	yes	load	load 123	alternate	Yes	No	Yes	No			Default - Matching Day pair	Alt



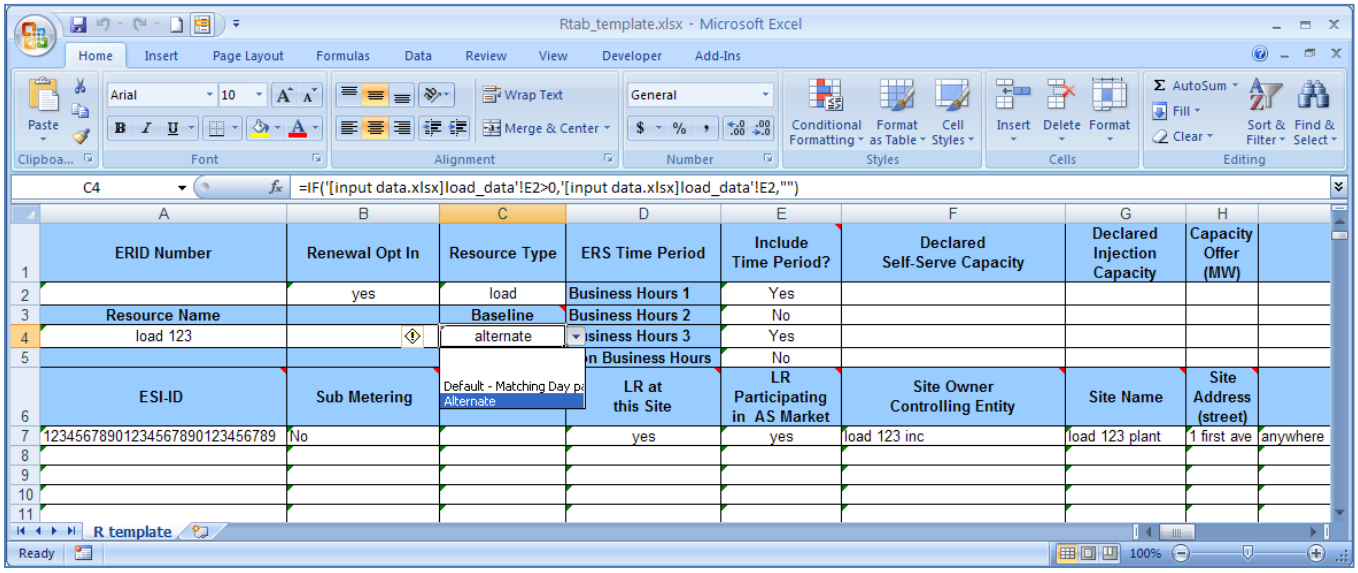
esiid	sub_metering	unique_meter_id	load_resource	lr_in_as	site_ownr	site_name	street	city	zipcode	tspdsp_i	tspdsp_ia_ye
12345678901234567890123456789	No		yes	yes	load 123 inc	load 123 plant	1 first ave	anywhere		yes	2002

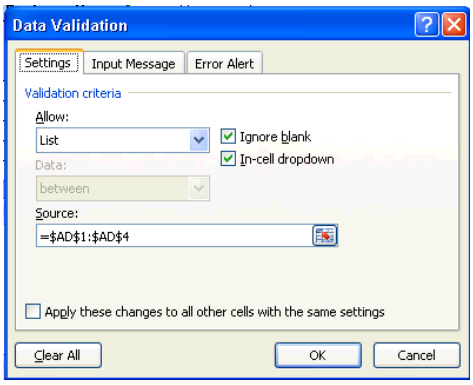


A portion of the 'rtab_template.xlsx' workbook is shown; the selected cell, A7, has a link to cell A2 on the 'site_data' tab in the 'input data.xlsx' workbook. The link is set up as an IF statement so that if the cell being linked to is populated, the value of the A2 cell is used, otherwise the cell is set to null. All cells in the 'rtab_template' in rows 7 and beyond are similar links to the appropriate cells on the 'site_data' tab of input data.xlsx.

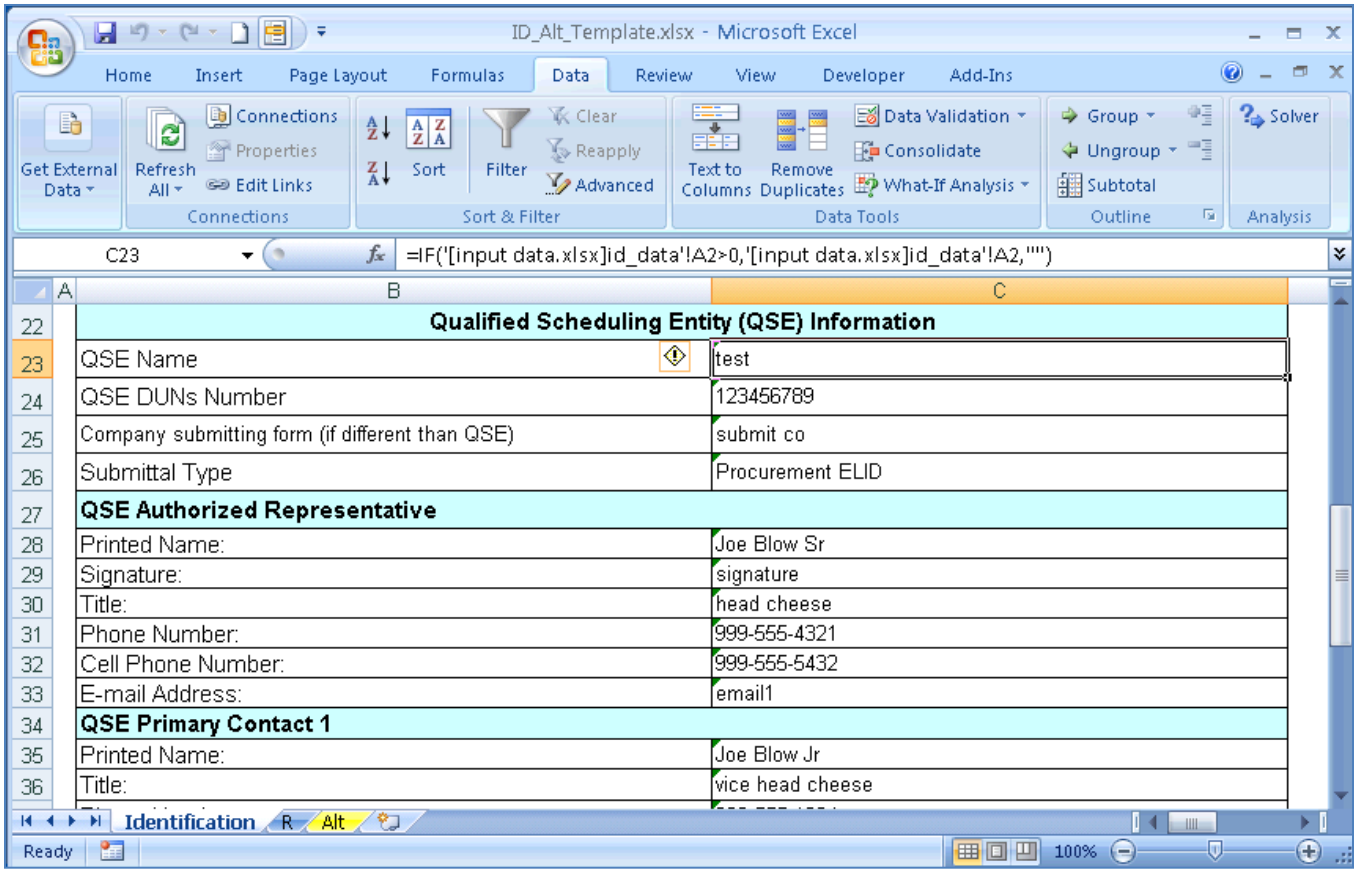


Cells on the template that need to be populated by SAS in rows 1 – 5 are similar links but connect to the 'load_data' tab. The baseline cell, C4, can be directly populated, but also has an associated drop-down list in cells AD1 – AD4, which also are links to the 'input data.xlsx' workbook. Using this technique, SAS can control the items showing up in the drop down list; for this application it is the way ERCOT uses to control baseline selection options that are available to the QSE for each resource.





A portion of the 'id_alt_template.xlsx workbook' is shown; the selected cell, C23, has a link to cell A2 on the 'id_data' tab in the 'input data.xlsx' workbook. As with the 'rtab_template', the link is set up as an IF statement so that if the cell being linked to is populated, the value of the A2 cell is used, otherwise the cell is set to null. All cells containing QSE provided data in column C between rows 23 - 51 are links to the appropriate cells on the 'id_data' tab of input data.xlsx. The submittal type cell, C26, can be directly populated, but also has an associated drop-down list in cells AC1 – AC10, which also are links to the input data.xlsx workbook. Using this technique, SAS can control the items showing up in the drop down list; for this application it is the way ERCOT uses to control the allowed submission type to be used by the QSE.



Using the three workbooks with appropriate links and opened on the PC running SAS, the code is now ready to move data from SAS datasets out to Excel. The libname, shown below, sets up a connection to the 'id_inputs' named range on the 'input_data.xlsx' workbook; this is the output destination used by SAS for the data. The proc datasets deletes all rows from the 'id_inputs' range … SAS requires that the range be blank if it is being used for output. The data step outputs the required data to the named range and, since the 'id_alt_template_copy.xlsx' workbook is open and has links on its 'Identification' tab to the cells in the 'id_inputs' range, the values of those links are updated by this data step.

It should be noted that when the Excel libname engine is used to write data out to an Excel workbook, it automatically includes a title row first and then writes variable values. To work around this limitation, this code used the 'input_data.xlsx' workbook as the output destination rather than outputting directly to the final workbook.

SAS - [Step 15 DDE_Code_Write out ok erids.sas]

```
454         libname mylib "&local_drive.\&templatepath.\input data.xlsx";
455
456         proc datasets lib = mylib noprint;
457             delete id_inputs;
458         quit;
459
460         data mylib.id_inputs;
461
462             keep qsename qseduns id_info1 - id_info23 sub_type1 sub_type2;
463
464             set okay_ers_inputs_1 (where = (qseduns = "&qseduns.")); 
465             id_info2 = '';
466             sub_type1 = '10m Procurement Offer';
467             sub_type2 = '30m Procurement Offer';
468             output;
469             stop;
470
471         run;
```

Output - (Untitled) | Log - (Untitled) | Step 15 DDE_Code...

D:\test Ln 433, Col 1

The next data step, shown below, issues DDE commands to activate various tabs in the 'id_alt_template_copy.xlsx' workbook and to run Excel macros that are stored in 'personal.xls'. Initially, SAS activates the 'Identification' tab and runs a macro called 'copy_paste_values_id_tab', which is shown below. The macro does a 'copy – paste special values' on the AC1 – AC2 range and thus removes the links in the drop down list for the submission type. Then the macro does a 'copy – paste special values' on the B22 – C51 range and thus removes the remaining links on the 'Identification' tab.

SAS then successively activates the 'Exceptions' tab, the 'Availability_Results' tab and the 'Baseline_Info' tab on the 'id_alt_template_copy.xlsx' workbook; and then runs two additional Excel macros, 'expand_cols' and 'clear_lines', on each tab. These three tabs contain summary level information on the resources submitted by the QSE on the workbook and were written by proc export as described above. The macros, shown below, 'auto-fit' all the columns on the activated tab, and clear all cell outlines on that tab. The first of these macros eliminates the annoying task of expanding columns in a workbook exported by SAS, and the second macro eliminates mysterious cell outlines that appear somewhat randomly (as far as I can tell) on these tabs.

I should point out that I am not a VBA programmer, and that all the macros described in this paper were produced by recording keystrokes that accomplished the desired tasks. In some case I have done minor editing primarily to clean up and simplify the macros.

SAS - [Step 15 DDE_Code_Write out ok erids.sas]

File Edit View Tools Run Solutions Window Help

```
478         data _null_;
479             file ddecmds;
480             put '[workbook.activate("Identification")]';
481             put '[run("PERSONAL.XLS!copy_paste_values_id_tab")]';
482             put '[workbook.activate("Exceptions")]';
483             put '[run("PERSONAL.XLS!expand_cols")]';
484             put '[run("PERSONAL.XLS!clear_lines")]';
485             put '[workbook.activate("Availabilty_Results")]';
486             put '[run("PERSONAL.XLS!expand_cols")]';
487             put '[run("PERSONAL.XLS!clear_lines")]';
488             put '[workbook.activate("Baseline_Info")]';
489             put '[run("PERSONAL.XLS!expand_cols")]';
490             put '[run("PERSONAL.XLS!clear_lines")]';
491             stop;
492         run;
```

Output - (Untitled) | Log - (Untitled) | Step 15 DDE_Code...

D:\test Ln 433, Col 1

Microsoft Visual Basic - PERSONAL.XLS - [Module1 (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 98, Col 1

(General) copy_paste_values_id_tab

```
Sub copy_paste_values_id_tab()
'
' copy_paste_values_id_tab Macro
'
'

     Range("ac1:AC2").Select
    Application.CutCopyMode = False
    Selection.Copy
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Range("aC1").Select

    Range("B22:AC51").Select
    Application.CutCopyMode = False
    Selection.Copy
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Range("C22").Select
End Sub
```

```
Sub expand_cols()
'
' expand_cols Macro
'
'

    Cells.Select
    Cells.EntireColumn.AutoFit
    Range("A1").Select
End Sub
```

```
Sub clear_lines()
'
' clear_lines Macro
'


'
    Cells.Select
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    Selection.Borders(xlEdgeLeft).LineStyle = xlNone
    Selection.Borders(xlEdgeTop).LineStyle = xlNone
    Selection.Borders(xlEdgeBottom).LineStyle = xlNone
    Selection.Borders(xlEdgeRight).LineStyle = xlNone
    Selection.Borders(xlInsideVertical).LineStyle = xlNone
    Selection.Borders(xlInsideHorizontal).LineStyle = xlNone
    Range("A1").Select
End Sub
```

The next two sections of code deal, in a similar way, with outputting resource level and site level data; the first section outputs to the 'Alt' tab and the second outputs to standard resource tabs. Since the code is similar, the description below will focus on the standard resource tab.

The code to write out standard resource tabs is, again, embedded in a macro do loop to cycle through all the tabs that needed to be written for the QSE. The first data step creates macro variables to specify the tab being output; as above, the proc datasets clears the Excel range, 'load_inputs', prior to the output, and the second data step outputs resource level to the range. The steps are repeated for the site level data; an important difference for site data is the output will consist of one or more rows based on the number of sites associated with the resource.

SAS - [Step 15 DDE_Code_Write out ok erids.sas *]

```
563     %if &numerids. > 0 %then %do j = 1 %to &numerids;
564
565       data _null_;
566         obsnum = input("&j",4.);
567         set qse_erid_list_1 point = obsnum;
568         call symputx("eridnum",put(eridnum,20.));
569         call symputx("tab_name", tab_name);
570         stop;
571       run;
572
573       proc datasets lib = mylib noprint;
574         delete load_inputs;
575       quit;
576
577       data mylib.load_inputs;
578
579         keep eridnum renewal_opt_in resource_type resource_name baseline
580           analyze_offer1 - analyze_offer4 blout1 - blout4 load ss_gen non_ss_gen;
581
582         set qse_erid_list_1 (where = (eridnum = &eridnum.));
583
584         output;
585         stop;
586
587       run;
588
589       proc datasets lib = mylib noprint;
590         delete site_inputs;
591       quit;
592
593       data mylib.site_inputs;
594
595         keep esiid sub_metering unique_meter_id load_resource lr_in_as site_ownr site_name street
596           city zipcode tspdsp_ia tspdsp_ia_year tspdsp_name tspdsp_duns noie_area wsale_meter
597           priv_net onsite_gen gen_type load_desc;
598
599         set okay_ers_rtab_inputs (where = (eridnum = &eridnum.));
600         nobs + 1;
601         call symputx("nobs", nobs);
602
603       run;
```

Output - (Untitled) | Log - (Untitled) | Step 15 DDE_Code...

Double click to change current folder | O:\craish\My Documents\sas user conf | Ln 589, Col 1

The next data step, shown below, issues DDE commands to activate the tab being processed in either the 'id_template_copy.xlsx' or the 'id_alt_template_copy.xlsx' workbooks and to run Excel macros that also are stored in 'personal.xls'. Initially, SAS activates the tab and runs a macro called 'copy_paste_rtab_template_to_id_template' or 'copy_paste_rtab_template_to_id_alt_template', which is shown below. The macro does a 'copy – paste' from 'rtab_template.xlsx' (only one tab in this workbook) to the appropriate tab (which was previously exported by SAS using the 'fill' data set) on either the 'id_template_copy.xlsx' or the 'id_alt_template_copy.xlsx' workbook. This copies all the fixed information and all the links to the destination workbook.

SAS then runs the Excel macro, 'copy_paste_values_alt_tab' shown below to eliminate all the links on the destination tab. The final set of macros deletes unneeded formats and rows from the destination tab and runs the 'expand_cols' macro described earlier. We have also installed the Excel add-in called 'clear excess formats' and frequently run it manually on the final workbooks … many of ours become quite large, and the add-in reduces them size by large amounts.

SAS - [Step 15 DDE_Code_Write out ok erids.sas]

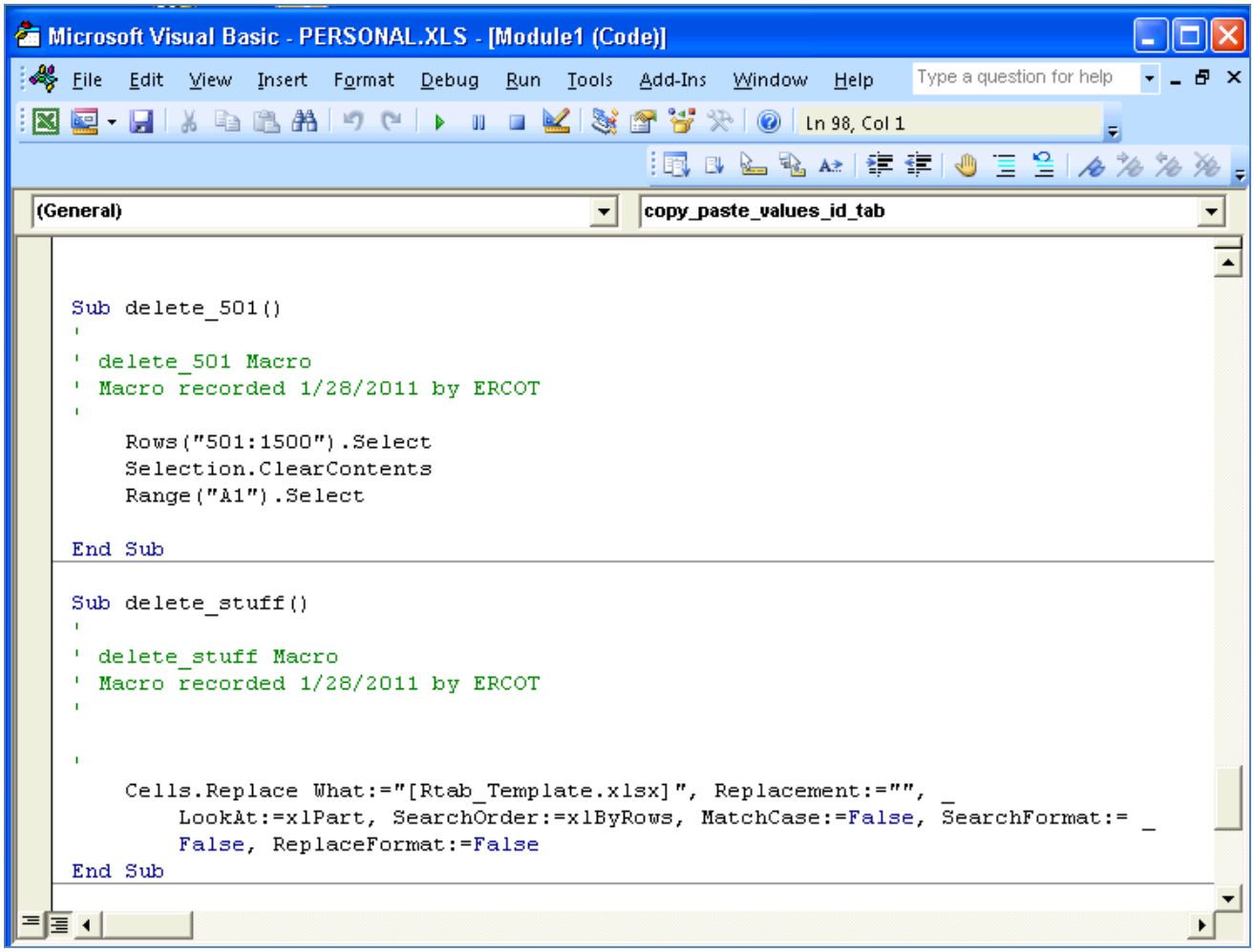
```
631             data _null_;
632                 file ddecmds;
633                 put '[workbook.activate("'"&tab_name."'")]';
634                 if "ID_Alt_Template" = "&ID_Template."
635                     then put '[run("PERSONAL.XLS!copy_paste_rtab_template_to_id_alt_template")]';
636                 else put '[run("PERSONAL.XLS!copy_paste_rtab_template_to_id_template")]';
637                 put '[run("PERSONAL.XLS!copy_paste_values_alt_tab")]';
638                 put '[run("PERSONAL.XLS!delete_stuff")]';
639                 if &nobs. < 40 then put '[run("PERSONAL.XLS!delete_51")]';
640                 else if &nobs. < 75 then put '[run("PERSONAL.XLS!delete_101")]';
641                 else if &nobs. < 250 then put '[run("PERSONAL.XLS!delete_301")]';
642                 else if &nobs. < 400 then put '[run("PERSONAL.XLS!delete_501")]';
643                 put '[run("PERSONAL.XLS!expand_cols")]';
644                 stop;
645
646             run;
```

Output - (Untitled) | Log - (Untitled) | Step 15 DDE_Code...

D:\test | Ln 645, Col 17

```
Sub copy_paste_rtab_template_to_id_alt_template()
'
' copy_paste_rtab_template_to_id_alt_template macro
'
'
    Windows("Rtab_Template.xlsx").Activate
    Cells.Select
    Selection.Copy
    Windows("ID_Alt_Template_copy.xlsx").Activate
    ActiveSheet.Paste
    Range("A1").Select
End Sub
```

```
Sub copy_paste_values_alt_tab()
'
' copy_paste_values_alt_tab Macro
'
'
    Range("A1:AF1500").Select
    Selection.Copy
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Application.CutCopyMode = False
    Selection.NumberFormat = "@"
    Range("A2").Select
End Sub
```



The final section of code issues DDE commands to Excel to save and close the three open workbooks. The currently active workbook at the beginning of this code is either 'id_template_copy.xlsx' or 'id_alt_template_copy.xlsx', so the first 'save as' command applies to this workbook. The 'save as' command is constructed in the required format using macro variables for the path and filename. Double quotes are used inside the 'save as' command to allow the macro variables to be resolved, the single quotes are needed by the syntax of the DDE command, and the %unquote unmasks the single quotes created by the %str function to turn the single quotes into values the SAS compiler recognizes. The 'sleep' commands are all used to give time for the Excel command to complete before the next command is issued by SAS. SAS then issues a DDE close command to 'close' the currently active workbook.

The next two 'save as' and 'close' commands are issued to close the remaining two open workbooks, and, when the second one is closed, a 'quit' command is issued to close down Excel. These commands are issued to allow SAS to deal directly with closing Excel with no user intervention required inside Excel; a simple close command will make Excel open a dialog box asking the user whether the workbook should be saved. Since ERCOT frequently runs large numbers of QSEs in a single batch, this smoothes out the process considerably.

The final data step issues system commands to delete the two dummy workbooks that created by the previous step as well as either the 'id_template_copy.xlsx' or 'id_alt_template_copy.xlsx' workbooks. This is done as a clean-up to delete workbooks that are no longer needed and to remove potential conflicts with subsequent iterations inside the 'run_qse' macro.

SAS - [Step 15 DDE_Code_Write out ok erids.sas]

```
655         data _null_;
656             file ddecmds;
657
658             put %unquote(%str(%'[SAVE.AS("&local_drive.\&eridpath.\RPT.00013012.&qseduns_fname..erids_okay.xlsx")]%'));
659             rc = sleep(2);
660             put '[close]';
661             rc = sleep(2);
662
663             put %unquote(%str(%'[SAVE.AS("&local_drive.\&templatepath.\dummy1.xlsx")]%'));
664             rc = sleep(2);
665             put '[close]';
666             rc = sleep(2);
667
668             put %unquote(%str(%'[SAVE.AS("&local_drive.\&templatepath.\dummy2.xlsx")]%'));
669             rc = sleep(2);
670             put '[close]';
671             rc = sleep(2);
672             put '[quit()]';
673             stop;
674         run;
675
676             /* delete template copy and two dummy files between qses */
677
678         data _null_;
679             length command $ 200;
680             command="&local_drive.";
681             call system(command);
682             command="cd\&templatepath.\";
683             call system(command);
684             command="del &ID_Template._copy.xlsx";
685             call system(command);
686             command="del dummy1.xlsx";
687             call system(command);
688             command="del dummy2.xlsx";
689             call system(command);
690
691         run;
```

Output - (Untitled) | Log - (Untitled) | Step 15 DDE_Code...

D:\test | Ln 645, Col 17

Some Additional Tips

When we first started running this code all inputs and outputs to Excel were run using DDE commands; use of the Excel libname engine greatly speeds up and improves the accuracy and reliability of the processes. Our latest iteration of the code, as described, eliminates as much as possible use of DDE commands.

From the beginning we have always run a verify step after writing to Excel workbooks with SAS. Though communication errors seem to have been eliminated with this version of the code, we still, from time to time, find other 'logic' errors when we read the workbook back in with code similar to that shown in the beginning of this paper and running proc compare to compare the resulting SAS dataset to the one used for the output code.

The code is fun to watch because both SAS and Excel are actively doing things on your desktop; it's helpful to keep an eye on the Excel, if the code appears to be hung up. In many cases it's because Excel has encountered a condition, usually an error, requiring user input via a dialog box. In these cases, you need to cancel submitted code in SAS and also shut down Excel manually.

Not surprisingly, diagnosing errors, particularly distinguishing between SAS errors and Excel macro errors is a bit mysterious … the error messages are usually very uninformative.

APPENDIX A

Complete Read-in Code

```
options symbolgen mlogic mprint linesize = 100 msglevel = i;


%global version_error;
%let version = 12.3;
%let contract_period = 2012 OctJan13;
%let version_error = No;
%let eridpath = d:\test;
%let offerpath = d:\test;
%let reportpath = d:\test;


libname perm "d:\test";


data fnames_list (Drop = rc fileid filecount i) ;

  length filename $ 300  ;

  rc = filename("Input", "&offerpath.");
     fileid = dopen("Input");
     filecount = dnum(fileid);
  DO i = 1 to filecount;

    filename = dread(fileid,i);
    if upcase(scan(filename,-1,'.')) in('XLS ', 'XLSM', 'XLSX') then Output;

  end;
     rc = DClose(fileid);

run;

/* create a macro variable of the number of filenames in the folder */
data _null_;

  set FNames_List nobs = numfiles;
  call symputx("numfiles",put(numfiles,4.));
  stop;

run;


%let i = 2;
  %do i = 1 %to &numfiles.;


/* read each workbook and all R-tabs and accumulate in dataset */
%macro read_work_book;

  %let first = 1;

  %do i = 2 %to 2;

    data _null_ ;
```

```
      obsnum = input("&i.",4.);
      set fnames_List point = obsnum;
      call symputx("filename",put(filename,$300.));
      stop;

    run;

    libname mylib "&offerpath.\&filename." mixed = yes header = no;

      ods trace on;
      ods output "Library Members" = tabnames;

      proc datasets library = mylib;
      quit;

      ods output close;
      ods listing;

    %let numbad = 0;
      data tab_names_1 (keep = filename tab_name)    bad_tab_names (keep = filename
tab_name);

      length tab_name $ 35 filename $ 300;
      set tabnames;

      filename = "&filename.";
      if index(name, "'") > 0 then tab_name =  "'" || trim(tranwrd(name, "'", "'")) ||
"'";
      else tab_name = name;

      if length(trim(tab_name)) >= 4 then do;
        if index(substr(tab_name, 2, length(trim(tab_name)) - 2), "'") > 0 &
          index(substr(tab_name, 2, length(trim(tab_name)) - 2), "&") > 0 then do;
          output bad_tab_names;
              call symputx("numbad",1);
        end;
          else do;
          if upcase(substr(tab_name,1,4))
            in('AVAI', 'BASE', 'EXCE', 'ID_I', 'IDEN', 'R$', 'SHEE')
            | index(tab_name,'Print_Area') > 0 then delete;
          else output tab_names_1;
        end;
      end;
        else if upcase(tab_name) ^= 'R$' then output tab_names_1;

      run;

    %if &numbad. = 1 %then %do;
      %put ;
      %put %str(ERROR: !!!!!!!!!!!!! Tab Name with Invalid Characters in Workbook
!!!!!!!!!!!!!!!!!!!  );
      %put ;
    %end;

    %let dset = mylib.'identification$'n ;
    %let dsid2 = %sysfunc(open(&dset.));
    %if &dsid2. %then %do;
      %let rc = %sysfunc(close(&dsid2.));
    %end;
    %let rc=%sysfunc(close(&dsid2.));

    data miss_id_tab;
```

```
  length filename $ 300;

  filename = "";

run;

%if &dsid2. = 0 %then %do;
  %put ;
  %put %str(ERROR: !!!!!!!!!!!!! Identification Tab Missing !!!!!!!!!!!!!!!!!!  );
  %put ;

  data miss_id_tab;

    length filename $ 300;

    filename = "&filename.";

  run;
%end;

%else %do;

  data version (keep = version_id temp_merge_var);
    length version_id $ 5;
    set mylib.'identification$'n;
    version_id = compress(f27);
    temp_merge_var = 1;
    output;
    stop;
  run;

  data identification (keep = qsename qseduns temp_merge_var id_info1 - id_info23);

    length qsename qseduns id_info1 -  id_info23 $64.;
    array id_info{23} $ id_info1 -  id_info23;
    retain qsename qseduns id_info1 -  id_info23;

    set mylib.'id_input_area'n ;

    if _n_ = 1 then qsename = f2;
    else if _n_ = 2 then qseduns = f2;
    if (_n_ >= 3 & _n_ <= 4) | (_n_ >= 6 & _n_ <= 11) | (_n_ >= 13 & _n_ <= 17)
      | (_n_ >= 19 & _n_ <= 23) | (_n_ >= 25 & _n_ <= 29) then do;
      i + 1;
      id_info{i} = f2;
      if i = 2 then id_info2 = upcase(id_info2);
      temp_merge_var = 1;
      if i = 23 then output;
    end;
  run;

  data identification_1;
    merge identification version;
    by temp_merge_var;
  run;

  data _null_;
    set tab_names_1 nobs = numtabs;
    put numtabs=;
    call symputx("numtabs",put(numtabs,4.));
    stop;

  run;
```

```
      %do j = 1 %to &numtabs.;

        data tab_names_2 (keep = tab_name temp_merge_var) ;

          length tab_name_quote $ 300;

          obsnum = input("&j.",4.);
          set tab_names_1 point = obsnum;

              if substr(tab_name, 1, 1) = "'" & substr(tab_name, length(trim(tab_name)),
1) = "'"
            & index(substr(tab_name, 2, length(trim(tab_name)) - 2), "'") > 0 then do;
                tab_name_quote = 'sheet = "' || trim(put(tab_name,$300.)) || '"';
            call symputx("sheet", tab_name_quote);
          end;

              else if substr(tab_name, 1, 1) = "'" & substr(tab_name,
length(trim(tab_name)), 1) = "'" then do;
                tab_name_quote = "sheet = " || trim(put(tab_name,$300.));
            call symputx("sheet", tab_name_quote);
          end;

              else do;
                tab_name_quote = "sheet = '" || trim(put(tab_name,$300.)) || "'";
                  call symputx("sheet", tab_name_quote);
          end;

              temp_merge_var = 1;
              output;
          stop;

        run;


            proc import datafile="&eridpath.\&filename." replace out = import_resource
dbms = excel;
          mixed = yes;
          getnames = no;
          scantext = yes;
          &sheet.;
        run;

            data import_resource_1;

                  set import_resource;

                  temp_merge_var = 1;

            run;

        data resource (keep = version_resource filename tab_name renewal_opt_in
resource_type
            resource_name baseline analyze_offer1 - analyze_offer4
            esiid sub_metering unique_meter_id load_resource lr_in_as site_ownr site_name
            street city zipcode tspdsp_ia tspdsp_ia_year tspdsp_name tspdsp_duns
            noie_area wsale_meter priv_net onsite_gen gen_type load_desc
            temp_merge_var eridnum rundate);

          length version_resource $ 5 filename $ 300 tab_name $ 35
            resource_type resource_name  baseline $ 64 renewal_opt_in $ 3
            esiid $ 64 sub_metering $ 3 unique_meter_id $ 64 load_resource lr_in_as $ 3
            site_ownr site_name $ 64 street city $64 zipcode $5. tspdsp_ia $ 3
```

```
            tspdsp_ia_year  tspdsp_name tspdsp_duns $64. noie_area wsale_meter
            priv_net onsite_gen $3. gen_type $30. load_desc $256.
            analyze_offer1 - analyze_offer4 $ 3;

          format rundate datetime20.;

          retain version_resource temp_merge_var filename tab_name renewal_opt_in
resource_type
            resource_name baseline analyze_offer1 - analyze_offer4 eridnum rundate;

          merge import_resource_1 tab_names_2;
              by temp_merge_var;

              if tab_name = 'Alt' then baseline = 'ALTERNATE';
              if _n_ = 1 then do;
                temp_merge_var = 1;
                        *tab_name = substr("&sheet.", 9, 1);
                tab_name = translate(tab_name, " ", "$");
                filename = "&filename.";
                rundate = datetime();
                eridnum = round(100 * (rundate - "01jun2009:01:00:00"dt),1);
                version_resource = f27;
              end;
              else if _n_ >= 2 & _n_ <= 5 & tab_name ^= 'Alt' then do;
                if _n_ = 2 then do;
                  renewal_opt_in = '';
                  resource_type = f3;
                  analyze_offer1 = upcase(f5);
                end;
                else if _n_ = 3 then analyze_offer2 = upcase(f5);
                else if _n_ = 4 then do;
                  resource_name = f1;
                  if resource_name = '' then resource_name = 'Missing Resource Name';
                  baseline = upcase(f3);
                  analyze_offer3 = upcase(f5);
                end;
                else if _n_ = 5 then analyze_offer4 = upcase(f5);
              end;
              else if (tab_name = 'Alt' & _n_ >= 4) | (tab_name ^= 'Alt' & _n_ >= 7) then
do;
                esiid = compress(f1, "' ");
                sub_metering = upcase(f2);
                unique_meter_id = f3;
                if esiid > '' & unique_meter_id = '' & sub_metering = ''
                  then sub_metering = 'NO';
                load_resource = upcase(f4);
                lr_in_as = upcase(f5);
                site_ownr = f6;
                site_name = f7;
                street = f8;
                city = f9;
                zipcode = f10;
                tspdsp_ia = upcase(f11);
                tspdsp_ia_year = f12;
                tspdsp_name = f13;
                tspdsp_duns = f14;
                noie_area = upcase(f15);
                wsale_meter = upcase(f16);
                priv_net = upcase(f17);
                onsite_gen = upcase(f18);
                gen_type = f19;
                load_desc = f20;
                if esiid > '' | unique_meter_id > '' then output;
```

```
                else if esiid = '' & unique_meter_id = ''
                  & (sub_metering > '' | unique_meter_id > '' | load_resource > '' |
lr_in_as > ''
                  | site_ownr > '' | site_name > '' | street > '' | city > '' | zipcode >
''
                  | tspdsp_ia > '' | tspdsp_ia_year > '' | tspdsp_name > '' | tspdsp_duns
> ''
                  | noie_area > '' | wsale_meter > '' | priv_net > '' | onsite_gen > ''
                  | gen_type > '' | load_desc > '') then
                  put /// 'esiid and umi both blank ' filename = tab_name = /// '';
              end;
            run;

        %let dset = work.resource;
        %let dsid3 = %sysfunc(open(&dset));

        %if &dsid3 %then %do;

          %let numloads =%sysfunc(attrn(&dsid3,NOBS));
          %let rc = %sysfunc(close(&dsid3));

        %end;

        %let rc=%sysfunc(close(&dsid3));
        %if &numloads. > 0 %then %do;

          data combine_id_resource bad_version (keep = qsename eridnum);

            merge identification_1 (in = in1) resource (in = in2);
            by temp_merge_var;

            if  ^in1 | ^in2 | compress(version_id) ^= "&version."
              | compress(version_resource) ^= "&version." then do;
              output bad_version;
              call symputx("version_error", "Yes");
            end;
            output combine_id_resource;
          run;

          proc sort data = bad_version nodupkey;
            by eridnum;
          run;

          proc sort data = combine_id_resource;
            by eridnum;
          run;

          data combine_id_resource_1 bad_version_1 (keep = qsename filename tab_name);
            merge combine_id_resource bad_version (in = in1);
            by eridnum;
            if ^in1 then output combine_id_resource_1;
            else if first.eridnum then output bad_version_1;
          run;

          %if &first. = 1 %then %do;

            data all_resources;
              set combine_id_resource_1;
            run;

            data all_bad_versions;
              set bad_version_1;
            run;
```

```
            data all_tab_names;
              set tab_names_1;
            run;

            data all_bad_tab_names;
              set bad_tab_names;
            run;

            data all_miss_id_tab;
              set miss_id_tab (where = (filename > ''));
            run;


            %let first = 0;

          %end;
          %else %do;

            proc append base = all_resources data = combine_id_resource_1;
            run;

            proc append base = all_bad_versions data = bad_version_1;
            run;

            proc append base = all_tab_names data = tab_names_1;
            run;


          %end;
        %end;
      %end;
    %end;

    proc append base = all_bad_tab_names data = bad_tab_names;
    run;

    proc append base = all_miss_id_tab data = miss_id_tab (where = (filename > ''));
    run;

  %end;

%mend;

%read_work_book;

libname mylib clear;




%macro check_errors;

  %let tabs_not_read = No;

  data all_tab_names_1;

    set all_tab_names;

    tab_name = translate(tab_name,' ', '$');

  run;
```

```
  proc sort data = all_tab_names_1 out = all_tab_names_2 nodupkey;
    by filename tab_name;
  run;

  proc sort data = all_resources out = all_resources_1;
    by filename tab_name;
  run;

  data tabs_not_read (keep = filename tab_name);

    merge all_tab_names_2 (in = in1) all_resources_1 (in = in2);
    by filename tab_name;

    if in1 & ^in2;
    call symputx("tabs_not_read", "Yes");

  run;

  %if &tabs_not_read. = Yes %then %do;

    %put;
    %put %str(ERROR: !!!!!!!!!!!!!!!!!!!! Tabs Not Read Error - Check 'tabs_not_read' sas
data set   !!!!!!!!!!!!!!!!!!!!!!!!!!!  );
    %put;

  %end;

  %let all_bad_tab_names = No;

  data _null_;

    set all_bad_tab_names ;

    if filename > '';
    call symputx("all_bad_tab_names", "Yes");

  run;



  %if &all_bad_tab_names. = Yes %then %do;
    %put ;
    %put %str(ERROR: !!!!!!!!!!!!! Tab Name(s) with Invalid Characters found in Workbook
- Check 'all_bad_tab_names' sas data set  !!!!!!!!!!!!!!!!!!  );
    %put ;
  %end;





  %if &version_error. = Yes %then %do;
    %put;
    %put %str(ERROR: !!!!!!!!!!!!!!!!!!!!! Submission Form Version Error Found - Check
'All_bad_versions' sas data set  !!!!!!!!!!!!!!!!!!!!!!!!!!  );
    %put;
  %end;
```

```
  %let all_miss_id_tab = No;

  data _null_;

    set all_miss_id_tab ;

    if filename > '';
    call symputx("all_miss_id_tab", "Yes");

  run;



  %if &all_miss_id_tab. = Yes %then %do;
    %put ;
    %put %str(ERROR: !!!!!!!!!!!!! Missing Identification Tab(s) found in Workbook -
Check 'all_miss_id_tab' sas data set  !!!!!!!!!!!!!!!!!!!  );
    %put ;
  %end;




%mend;

%check_errors;
```

APPENDIX B

Complete Write-out Code

```
options noxwait noxsync symbolgen mlogic mprint linesize = 200 msglevel = i;


%let contract_period = 2012 OctJan13;
%let local_drive = d:;          * enter the drive letter for the templates and output
path ;

/* path to write xls files and to see if they already exist for this batch */
%let eridpath = Emergency_Interruptible_Load\&contract_period.\00 resource
Identification\3 Outgoing erids\okay;
%let templatepath = Emergency_Interruptible_Load\&contract_period.\00 resource
Identification\DDE Templates;

libname perm "Z:\Emergency_Interruptible_Load\&contract_period.\00 resource
Identification\sas datasets";

  /***************************************************/
  /* create dataset of qsenames and count of erids  */
  /***************************************************/
data okay_ers_inputs (keep = eridnum qsename qseduns renewal_opt_in resource_name
    resource_type load_type baseline analyze_offer1 - analyze_offer4  blout1 - blout4
    load ss_gen non_ss_gen esiid sub_metering unique_meter_id load_resource
    lr_in_as site_ownr site_name site_name street city zipcode tspdsp_ia tspdsp_ia_year
    tspdsp_name tspdsp_duns noie_area wsale_meter priv_net onsite_gen gen_type load_desc
    id_info1 - id_info23);

  length eridnum 8 qsename qseduns $ 64 renewal_opt_in $ 3 resource_type resource_name
    baseline $ 64 analyze_offer1 - analyze_offer4 $ 3 blout1 - blout4 blout1 - blout4
load
    ss_gen non_ss_gen $ 30 esiid $ 64 sub_metering $ 3 unique_meter_id $ 64 load_resource
    lr_in_as $ 3 site_ownr site_name street city $ 64 zipcode $ 5 tspdsp_ia $ 3
tspdsp_ia_year
    tspdsp_name tspdsp_duns $ 64 noie_area wsale_meter priv_net onsite_gen $ 3 gen_type $
30
    load_desc $ 256 id_info1 - id_info23 $ 64;

  set perm.ers_inputs_okay_final;

  renewal_opt_in = '';
  load = 'Load';
  ss_gen = 'Self-Serving Gen';
  non_ss_gen= 'Non Self-Serving Gen';
  if bline1 = 1 then blout1 = 'DEFAULT-REGRESSION';
  if bline2 = 2 then blout2 = 'DEFAULT-MID 8 OF 10';
  if bline3 = 3 then blout3 = 'DEFAULT-MATCHING DAY PAIR';
  blout4 = 'ALTERNATE';
  if baseline = '' & bline1 = . & bline2 = . & bline3 = . then baseline = 'ALTERNATE';

run;

data okay_ers_inputs_1 (drop = temp_tab_name more i numbers letters);

  length tab_name $ 29 temp_tab_name $ 64;

  set okay_ers_inputs;
```

```
  if resource_name = '' then tab_name = 'Alt';

  else do;
    temp_tab_name = compress(resource_name, "'");
    temp_tab_name = compress(temp_tab_name, '"');
    temp_tab_name = translate(temp_tab_name,"                                ",
                        " !@#$%^&*()_+-={}|[]\:;<>?,./;");
    temp_tab_name = compbl(temp_tab_name);
    temp_tab_name = translate(temp_tab_name,"~", " ");
    more = 1;
    do i = 64 to 1 by - 1 while (more = 1);
      if substr(temp_tab_name,i,1) = '~' then substr(temp_tab_name,i,1) = ' ';
      else more = 0;
    end;
    tab_name = compbl(substr(translate(temp_tab_name, '_', '~'),1,26));
    if substr(tab_name, 26,1) = '_' then tab_name = substr(tab_name, 1, 25);
    letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
    numbers = '1234567890';
    if verify(substr(tab_name,1,3), letters) = 0
      & verify(compress(substr(tab_name,4)), numbers) = 0
      & substr(tab_name,4) > ''
      then tab_name = substr(tab_name,1,3) || '_' || substr(tab_name,4);
    else if verify(substr(tab_name,1,2), letters) = 0
      & verify(compress(substr(tab_name,3)), numbers) = 0
      & substr(tab_name,3) > ''
      then tab_name = substr(tab_name,1,2) || '_' || substr(tab_name,3);
    else if verify(substr(tab_name,1,1), letters) = 0
      & verify(compress(substr(tab_name,2)), numbers) = 0
      & substr(tab_name,2) > ''
      then tab_name = substr(tab_name,1,1) || '_' || substr(tab_name,2);
    if verify(substr(tab_name,1,1), letters) ^= 0 then tab_name = '_' || tab_name;
    tab_name = upcase(tab_name);
  end;

run;

proc sort data = okay_ers_inputs_1;
  by qseduns tab_name eridnum;
run;

data okay_ers_rtab_inputs okay_ers_alt_inputs ;

  drop count;

  set okay_ers_inputs_1;
  by qseduns tab_name eridnum;

  if tab_name ^= 'Alt' then do;
    if first.tab_name then count = 0;
    if first.eridnum then count + 1;
    if count > 1 & count < 100 then
      tab_name = translate(compbl(substr(tab_name,1,26) || put(count,2.)),'_',' ');
    output okay_ers_rtab_inputs;
  end;
  else output okay_ers_alt_inputs;

run;

proc sort data = okay_ers_rtab_inputs out = qse_erid_list nodupkey;
  by qseduns tab_name eridnum;
run;

proc freq data = okay_ers_inputs_1 noprint;
```

```
  tables qsename * qseduns  / out = qse_list;
run;

  /*******************************************/
  /* create macro variable of number of qses  */
  /*******************************************/

%let numqses = 0;

data _null_;
  set qse_list nobs = numqses;
  call symputx("numqses",put(numqses,5.));
  stop;
run;

%macro run_qse;

  data fill;
    a = 1; b = 1; c = 1;
    if a > 1;
  run;

    /* loop through QSEs */
  %if &numqses. > 0 %then %do i = 1 %to &numqses.;

    data _null_;
      obsnum = input("&i",4.);
      set qse_list point = obsnum;
      call symputx("qseduns",put(qseduns,$64.));
      call symputx("qseduns_fname",put(input(qseduns,13.),z16.));
      stop;
    run;

      /* set macro numerids to the number of r tabs for qse being processed */

    %let numerids = 0;

    /* create a file of eridnums for this qse */
    data qse_erid_list_1 ;
      set qse_erid_list (where = (qseduns = "&qseduns."));
      numerids + 1;
      call  symputx("numerids",numerids);
    run;

    /* set macro variable for the template to be used (with or without and Alt tab) */

    %let ID_Template = ID_Template;

    data _null_;

      set okay_ers_alt_inputs (where = (qseduns = "&qseduns."));
      call symputx("ID_Template", "ID_Alt_Template");
      stop;
    run;

      /* copy ID_Template.xlsx into ID_Template_copy.xlsx */

    data _null_;
      length command $ 200;
      command="&local_drive.";
          call system(command);
          command="cd\&templatepath.\";
          call system(command);
```

```
  command="copy &ID_Template..xlsx &ID_Template._copy.xlsx";
      call system(command);
run;

%if &numerids. > 0 %then %do j = 1 %to &numerids.;


/* create tabs for all the R tab_names needed for this qse */
  data _null_;
    obsnum = input("&j",4.);
    set qse_erid_list_1 point = obsnum;
    call symputx("tab_name", tab_name);
    stop;
  run;

  proc export data = fill outfile =
    "&local_drive.\&templatepath.\&ID_Template._copy.xlsx";
    sheet="&tab_name.";
  run;

%end;

data messages_erids_okay ;
  set perm.messages_erids_okay;
  where qseduns = "&qseduns." ;
run;

proc sort data = messages_erids_okay ;
  by qsename resource_name tab_name eridnum esiid unique_meter_id;
run;

data rtabs_with_messages (keep = qsename resource_name tab_name eridnum)
  alttabs_with_messages
    (keep = qsename resource_name tab_name eridnum esiid unique_meter_id);

  set messages_erids_okay ;
  if message1 ^= '' |  message2 ^= '' |  message3 ^= '' |  message4 ^= ''
    |  message5 ^= '' |  message6 ^= '' |  message_n ^= ''
    |  message_s ^= '' |  message_u ^= '';
  if tab_name = 'Alt' then output alttabs_with_messages;
  else output rtabs_with_messages;
run;

proc sort data = rtabs_with_messages nodupkey;
  by qsename resource_name tab_name eridnum;
run;

proc sort data = alttabs_with_messages nodupkey;
  by qsename resource_name tab_name eridnum esiid unique_meter_id;
run;

data messages_rtabs_okay;

  length qsename qseduns $ 64 renewal_opt_in $ 3 resource_name resource_type
    load_type baseline esiid $ 64 sub_metering $ 3 unique_meter_id $ 64 load_resource
    lr_in_as $ 3 street city $ 64 zipcode $ 5 message1 - message6 message_n message_s
    message_u $ 200 site_ownr site_name $ 64 tspdsp_ia $ 3 tspdsp_ia_year tspdsp_name
    tspdsp_duns $ 64 analyze_offer1 - analyze_offer4 noie_area wsale_meter priv_net
    onsite_gen $ 3 gen_type $ 30 load_desc $ 256 filename $ 300 tab_name $ 30;

  merge messages_erids_okay alttabs_with_messages(in = in1);
  by qsename resource_name tab_name eridnum esiid unique_meter_id;
```

```
  if in1;
  renewal_opt_in = '';

run;

data messages_alts_okay;

  length qsename qseduns $ 64 renewal_opt_in $ 3 resource_name resource_type
    load_type baseline esiid $ 64 sub_metering $ 3 unique_meter_id $ 64 load_resource
    lr_in_as $ 3 street city $ 64 zipcode $ 5 message1 - message6 message_n message_s
    message_u $ 200 site_ownr site_name $ 64 tspdsp_ia $ 3 tspdsp_ia_year tspdsp_name
    tspdsp_duns $ 64 analyze_offer1 - analyze_offer4 noie_area wsale_meter priv_net
    onsite_gen $ 3 gen_type $ 30 load_desc $ 256 filename $ 300 tab_name $ 30;

  merge messages_erids_okay rtabs_with_messages(in = in1);
  by qsename resource_name tab_name eridnum;

  if in1;
  renewal_opt_in = '';

run;

data messages_erids_okay_1;

  length qsename qseduns $ 64 renewal_opt_in $ 3 resource_name resource_type
    load_type baseline esiid $ 64 sub_metering $ 3 unique_meter_id $ 64 load_resource
    lr_in_as $ 3 street city $ 64 zipcode $ 5 message1 - message6 message_n message_s
    message_u $ 200 site_ownr site_name $ 64 tspdsp_ia $ 3 tspdsp_ia_year tspdsp_name
    tspdsp_duns $ 64 analyze_offer1 - analyze_offer4 noie_area wsale_meter priv_net
    onsite_gen $ 3 gen_type $ 30 load_desc $ 256 filename $ 300 tab_name $ 30;

  set messages_rtabs_okay  messages_alts_okay;
  by qsename resource_name tab_name eridnum esiid unique_meter_id;

  if eridnum > .;

  /* output resource level messages */
  if first.resource_name | tab_name = 'Alt' then output;

  /* blanks out load level info on the esiid rows */
  else do;
    qsename = ''; qseduns = ''; resource_name = '';  baseline = '';
    message1 = '';
    output;
  end;
  /* insert blank line after last load */
  if last.eridnum & eridnum > . then do;
    qsename = ''; qseduns = ''; renewal_opt_in = ''; resource_name = '';
    resource_type = ''; load_type = ''; baseline = ''; esiid = ''; sub_metering = '';
    unique_meter_id = ''; load_resource = ''; lr_in_as = ''; street = ''; city = '';
    zipcode = ''; message1 = ''; message2 = ''; message3 = ''; message4 = '';
    message5 = ''; message6 = ''; message_n = ''; message_s = ''; message_u = '';
    site_ownr = ''; site_name = ''; tspdsp_ia = ''; tspdsp_ia_year = '';
    tspdsp_name = ''; tspdsp_duns = ''; analyze_offer1 = ''; analyze_offer2 = '';
    analyze_offer3 = ''; analyze_offer4 = ''; noie_area = ''; wsale_meter = '';
    priv_net = ''; onsite_gen = ''; gen_type = ''; load_desc = ''; eridnum = .;
    filename = ''; tab_name = '';
    output;
  end;
run;

proc export data = messages_erids_okay_1
  outfile = "&local_drive.\&templatepath.\&ID_Template._copy.xlsx" replace;
```

```
  Sheet = "Exceptions";
run;

/* writes out availability results to workbooks */

data erid_avail_results;
  length resource_name site_name $ 64;
  set perm.erid_avail_results;
  where qseduns = "&qseduns." ;
run;

proc sort data = erid_avail_results ;
  by qsename resource_name eridnum esiid unique_meter_id contract_period
    time_period;
run;

data erid_avail_results
  (keep = qsename resource_name esiid unique_meter_id site_name
    min_esiid_ct contract_period time_period min_mw mw_1st_pctile mw_95th_pctile
    avg_mw eridnum);

  length qsename resource_name site_name esiid unique_meter_id
    contract_period $ 64  time_period $12 offer_mw min_mw mw_1st_pctile
    mw_95th_pctile avg_mw min_esiid_ct eridnum 8 ;

  label qsename = "QSE Name" resource_name = "Resource Name"  esiid = "ESI ID"
    unique_meter_id = "Unique_Meter_Id" site_name = "Site Name"
    contract_period = "Analysis Months"  time_period = "Time Period"
    offer_mw = "Offer MW"   min_mw = "Minimum Base Load MW"
    min_esiid_ct = "Number of Loads Analyzed"
    min_mw = "Minimum MW"
    mw_1st_pctile = "1st Percentile Load MW"
    mw_95th_pctile = "95th Percentile Load MW" avg_mw = "Average Load MW";

  set erid_avail_results;
  by qsename resource_name eridnum esiid unique_meter_id contract_period
    time_period;

  output;

  /* make a blank row between each contract period */
  if last.contract_period then do;
    qsename = '' ; resource_name = ' '; esiid= ' '; unique_meter_id = '';
    site_name = ' '; contract_period = ' '; time_period = ' '; offer_mw = .;
    min_esiid_ct = .; mw_95th_pctile = .; avg_mw = .;  min_mw = .; mw_1st_pctile = .;
    eridnum = .; load_resource = ''; lr_in_as = '';

    output;
  end;
  /* make a blank row between each contract period */
  if last.resource_name then do;
    qsename = '' ; resource_name = ' '; esiid= ' '; unique_meter_id = '';
    site_name = ' '; contract_period = ' '; time_period = ' '; offer_mw = .;
    min_esiid_ct = .; mw_95th_pctile = .; avg_mw = .;  min_mw = .; mw_1st_pctile = .;
    eridnum = .; load_resource = ''; lr_in_as = '';
    output;
  end;
run;
proc export data = erid_avail_results
  outfile = "&local_drive.\&templatepath.\&ID_Template._copy.xlsx" replace;
  Sheet = "Availabilty_Results";
run;
```

```
/* writes out baseline results to workbooks */

  data compare_ranking_report;
    length resource_name site_name $ 64;
    set perm.compare_ranking_report;
    where qseduns = "&qseduns." ;
  run;

  proc sort data = compare_ranking_report;
    by qsename resource_name eridnum esiid unique_meter_id contract_period;
  run;

  data export_baseline_info
    (keep = qsename resource_name contract_period esiid unique_meter_id
      site_name RSquare MAPE P90_kW_Confidence P95_kW_Confidence
      P99_kW_Confidence Baseline eridnum);

    length qsename resource_name site_name esiid unique_meter_id contract_period
      Baseline $ 64 RSquare MAPE P90_kW_Confidence P95_kW_Confidence
      P99_kW_Confidence eridnum 8. ;

    set compare_ranking_report;
    by qsename resource_name eridnum esiid unique_meter_id  site_name contract_period;

    where qseduns = "&qseduns." ;

     /* inserts blank row after load level messages*/
    output;
    if last.contract_period then do;

      RSquare = .;
      MAPE = .;
      P90_kW_Confidence = .;
      P95_kW_Confidence = .;
      P99_kW_Confidence = .;
      Baseline = ' ';
      resource_name = ' ';
      qsename = '';
      contract_period = ' ';
      esiid = ' ';
      unique_meter_id = ' ';
      site_name = ' ';
      eridnum = .;
      qsenum = .;
      output;
    end;
  run;

  proc export data = export_baseline_info
    outfile = "&local_drive.\&templatepath.\&ID_Template._copy.xlsx" replace;
    Sheet = "Baseline_Info";
  run;

  x "'C:\Program Files\Microsoft Office\OFFICE12\excel.exe'";

  data _null_;
    rc = sleep(2);
    stop;
  run;

  filename ddecmds dde 'excel|system';

    /* open the input data.xlsx workbook */
```

```
    data _null_;
      rc = sleep(1);
      file ddecmds;
      put %unquote(%str(%'[open("&local_drive.\&templatepath.\input data.xlsx")]%'));
      stop;
    run;

    data _null_;
      rc = SLEEP(1);
      file ddecmds;
      put %unquote(%str(%'[open("&local_drive.\&templatepath.\Rtab_Template.xlsx")]%'));
    run;

    data _null_;
      rc = SLEEP(1);
      file ddecmds;
      put
%unquote(%str(%'[open("&local_drive.\&templatepath.\&ID_Template._copy.xlsx")]%'));
    run;

    /* identification information for the QSE being processed  */
    /* pull the input data needed from the permanent sas dataset */
    /* and output to the 'input data.xlsx' workbook in the 'id_inputs' range */
    /* the 'ID_Alt_Template_copy.xlsx' workbook which ends up being the saved qse
workbook */
    /* has links to the 'input data.xlsx' workbook */

    libname mylib "&local_drive.\&templatepath.\input data.xlsx";

    proc datasets lib = mylib noprint;
      delete id_inputs;
    quit;

    data mylib.id_inputs;

      keep qsename qseduns id_info1 - id_info23 sub_type1 sub_type2;

      set okay_ers_inputs_1 (where = (qseduns = "&qseduns."));
      id_info2 = '';
      sub_type1 = '10m Procurement Offer';
      sub_type2 = '30m Procurement Offer';
      output;
      stop;

    run;



    data _null_;
      file ddecmds;
      put '[workbook.activate("Identification")]';
      put '[run("PERSONAL.XLS!copy_paste_values_id_tab")]';
      put '[workbook.activate("Exceptions")]';
      put '[run("PERSONAL.XLS!expand_cols")]';
      put '[run("PERSONAL.XLS!clear_lines")]';
      put '[workbook.activate("Availabilty_Results")]';
      put '[run("PERSONAL.XLS!expand_cols")]';
      put '[run("PERSONAL.XLS!clear_lines")]';
      put '[workbook.activate("Baseline_Info")]';
      put '[run("PERSONAL.XLS!expand_cols")]';
      put '[run("PERSONAL.XLS!clear_lines")]';
      stop;
```

```
    run;

    %if &ID_Template. = ID_Alt_Template %then %do;

      proc datasets lib = mylib noprint;
        delete load_inputs;
      quit;

      data mylib.load_inputs;

        keep eridnum renewal_opt_in resource_type resource_name baseline
          analyze_offer1 - analyze_offer4 blout1 - blout4 load ss_gen non_ss_gen;

        set okay_ers_alt_inputs (where = (qseduns = "&qseduns."));

        output;
        stop;

      run;

      proc datasets lib = mylib noprint;
        delete site_inputs;
      quit;

      data mylib.site_inputs;

        keep esiid sub_metering unique_meter_id load_resource lr_in_as site_ownr
site_name
          street city zipcode tspdsp_ia tspdsp_ia_year tspdsp_name tspdsp_duns noie_area
          wsale_meter priv_net onsite_gen gen_type load_desc;

        set okay_ers_alt_inputs (where = (qseduns = "&qseduns."));
        output;

        nobs + 1;
        call symputx("nobs", nobs);

      run;

      data _null_;
        file ddecmds;
        put '[workbook.activate("Alt")]';
        put '[run("PERSONAL.XLS!copy_paste_values_alt_tab")]';
        if &nobs. < 40 then put '[run("PERSONAL.XLS!delete_51")]';
        else if &nobs. < 75 then put '[run("PERSONAL.XLS!delete_101")]';
        else if &nobs. < 250 then put '[run("PERSONAL.XLS!delete_301")]';
        else if &nobs. < 400 then put '[run("PERSONAL.XLS!delete_501")]';
        put '[run("PERSONAL.XLS!expand_cols")]';
        stop;
      run;

      proc datasets lib = mylib noprint;
        delete site_inputs;
      quit;

    %end;

    %if &numerids. > 0 %then %do j = 1 %to &numerids;

      data _null_;
        obsnum = input("&j",4.);
        set qse_erid_list_1 point = obsnum;
        call symputx("eridnum",put(eridnum,20.));
```

```
        call symputx("tab_name", tab_name);
        stop;
      run;

      /* delete contents of cells in the load_inputs range ... excel engine cannot */

      proc datasets lib = mylib noprint;
        delete load_inputs;
      quit;

      data mylib.load_inputs;

        keep eridnum renewal_opt_in resource_type resource_name baseline
          analyze_offer1 - analyze_offer4 blout1 - blout4 load ss_gen non_ss_gen;

        set qse_erid_list_1 (where = (eridnum = &eridnum.));

        output;
        stop;

      run;

      /* delete contents of cells in the load_inputs range ... excel engine cannot
replace values */

      proc datasets lib = mylib noprint;
        delete site_inputs;
      quit;

      data mylib.site_inputs;

        keep esiid sub_metering unique_meter_id load_resource lr_in_as site_ownr
site_name
          street city zipcode tspdsp_ia tspdsp_ia_year tspdsp_name tspdsp_duns noie_area
          wsale_meter priv_net onsite_gen gen_type load_desc;

        set okay_ers_rtab_inputs (where = (eridnum = &eridnum.));
        nobs + 1;
        call symputx("nobs", nobs);

      run;

      data _null_;
        file ddecmds;
        put '[workbook.activate("'"&tab_name."'")]';
        if "ID_Alt_Template" = "&ID_Template."
          then put '[run("PERSONAL.XLS!copy_paste_rtab_template_to_id_alt_template")]';
        else put '[run("PERSONAL.XLS!copy_paste_rtab_template_to_id_template")]';
        put '[run("PERSONAL.XLS!copy_paste_values_alt_tab")]';
        put '[run("PERSONAL.XLS!delete_stuff")]';
        if &nobs. < 40 then put '[run("PERSONAL.XLS!delete_51")]';
        else if &nobs. < 75 then put '[run("PERSONAL.XLS!delete_101")]';
        else if &nobs. < 250 then put '[run("PERSONAL.XLS!delete_301")]';
        else if &nobs. < 400 then put '[run("PERSONAL.XLS!delete_501")]';
        put '[run("PERSONAL.XLS!expand_cols")]';
        stop;

      run;
    %end;

    libname mylib clear;

      /* save and close the template copy workbook with an MIS compatible file name */
```

```
      /*   and save and close the two dummy files for the other two open workbooks */
      /*  (this avoids having to respond in excel to okay losing the changes) */

    data _null_;
      file ddecmds;

      put
%unquote(%str(%'[SAVE.AS("&local_drive.\&eridpath.\RPT.00013012.&qseduns_fname..erids_oka
y.xlsx")]%'));
      rc = sleep(2);
      put '[close]';
      rc = sleep(2);

      put %unquote(%str(%'[SAVE.AS("&local_drive.\&templatepath.\dummy1.xlsx")]%'));
      rc = sleep(2);
      put '[close]';
      rc = sleep(2);

      put %unquote(%str(%'[SAVE.AS("&local_drive.\&templatepath.\dummy2.xlsx")]%'));
      rc = sleep(2);
      put '[close]';
      rc = sleep(2);
      put '[quit()]';
      stop;
    run;

      /* delete template copy and two dummy files between qses */

    data _null_;
      length command $ 200;
      command="&local_drive.";
          call system(command);
          command="cd\&templatepath.\";
          call system(command);
      command="del &ID_Template._copy.xlsx";
          call system(command);
      command="del dummy1.xlsx";
          call system(command);
      command="del dummy2.xlsx";
          call system(command);

    run;

  %end;
%mend;
%run_qse;
```