Paper 192-25

The Anatomy of a Report: A Macro Driven Approach

Diane E. Brown, TEC Associates, Indianapolis, IN

ABSTRACT

This information system is a Web based GUI constructed using HTML, JAVA, and CGI. The GUI allows the user to design a report and enter data subset criteria. These parameters are passed to the SAS® System for execution. SAS queries an Oracle® data warehouse using a WHERE clause built from user defined subset criteria. The result of the query is then processed through the SAS application to generate custom HTML output and GIF graphics.

The focus of this paper is the SAS application that generates the majority of the tabular style reports. Users can define the report layout, sort order, summarization levels for the report body and total lines, define multiple total lines, and create new variables through assignment statements.

INTRODUCTION

This application was built for TTC, Inc., a division of Dynatech, LLC, a telecommunications systems and software company. TTC is a leading provider of network performance management products that include systems, software, and services used to manage the quality, performance, availability, and reliability of telecommunications service providers' networks. These products enable telecommunications service providers to improve quality of service, increase productivity, and lower operating expenses. The company is headquartered in Germantown, MD, and has product development and support facilities in Vancouver, British Columbia, Terre Haute, IN, and Fairfax, VA. In addition, TTC has sales offices worldwide. Press releases and other information about TTC and its products and services are available at http://www.ttc.com.

Their application, TDC&E (Traffic Data Collection & Engineering), provides a complete service assurance package; including analysis of traffic data for equipment servicing, trunk forecasting, load balance, toll separations, customer studies, service level exceptions and regulatory requirements. Telecommunications industry requirements for the determination of workload busy times and optimum hardware configuration are supported. Busy determination supports the methods, bouncing busy, peak busy, predetermined busy, average busy, and time consistent busy. Optimum hardware calculations are based on the Erlang B, Erlang C, and Poisson distributions.

TDC&E TECHNICAL OVERVIEW

Telecommunications activity data is collected and stored in an Oracle Data Warehouse on a Unix platform. A Web based GUI written using HTML, JAVA, and CGI, gives the analyst the ability to design a report and store the specifications in Oracle tables. The report design defines the report layout, selects Oracle columns, derives new variables, defines sort order, report totals, and the variables appropriate for data subset. Once the report is defined, a less sophisticated user can complete the desired subset criteria and generate the report. Reports can be generated interactively or ran in a scheduled production environment.

The report definition and subset values are passed to SAS via an INPUT file containing defined parameters. These parameters are converted to SAS macro variables and a WHERE clause. The macro variables are used to build the requested report layout, sort order, new variables, total lines, and specific telecommunications procedures. Telecommunications procedures include the determination of workload busy times and optimum hardware requirements. The WHERE clause is used to subset Oracle tables.

Both tabular and graphic reports can be produced. Tabular output is in HTML format, along with optional text output and delimited file output. The HTML output includes a custom HTML macro and SAS Institute's Web Publishing Tool for PROC TABULATE. The text output is generated using PROC PRINT. Delimited output is generated through a custom macro. Graphic output is stored as GIF files. All output is stored in the OUTPUT directory with the same file name as the corresponding files in the INPUT directory.

The underlying SAS application is heavily based in macros and include libraries with approximately 87 source code modules. The driving source code consists of 5 tabular models, 3 graphic models, and 14 custom reports. Most reports are generated through the use of models and very specific reports are written as custom. The tabular model uses the custom HTML macro and the tabulate model uses the Web Publishing Tool for PROC TABULATE. The 3 other tabular models support telecommunications specific requirements. The 3 graphics models include plot, bar, and pie.

SAS products used for this application include Base SAS®, SAS/ACCESS® to Oracle, and SAS/GRAPH®.

TEC Associates Indianapolis, IN (317) 823-0642 diane_b@tecassociates.com

SAMPLE REPORTS

This is an example of output from the tabular model.

SUGI TELECOM

TEC Associates
Indianapolis, Indiana
September 25, 1996 through September 27, 1996

Trunk Group **120 4623**

Engineering Method **erlang b**
Engineering Type **c**
Grade of Service **0.01**

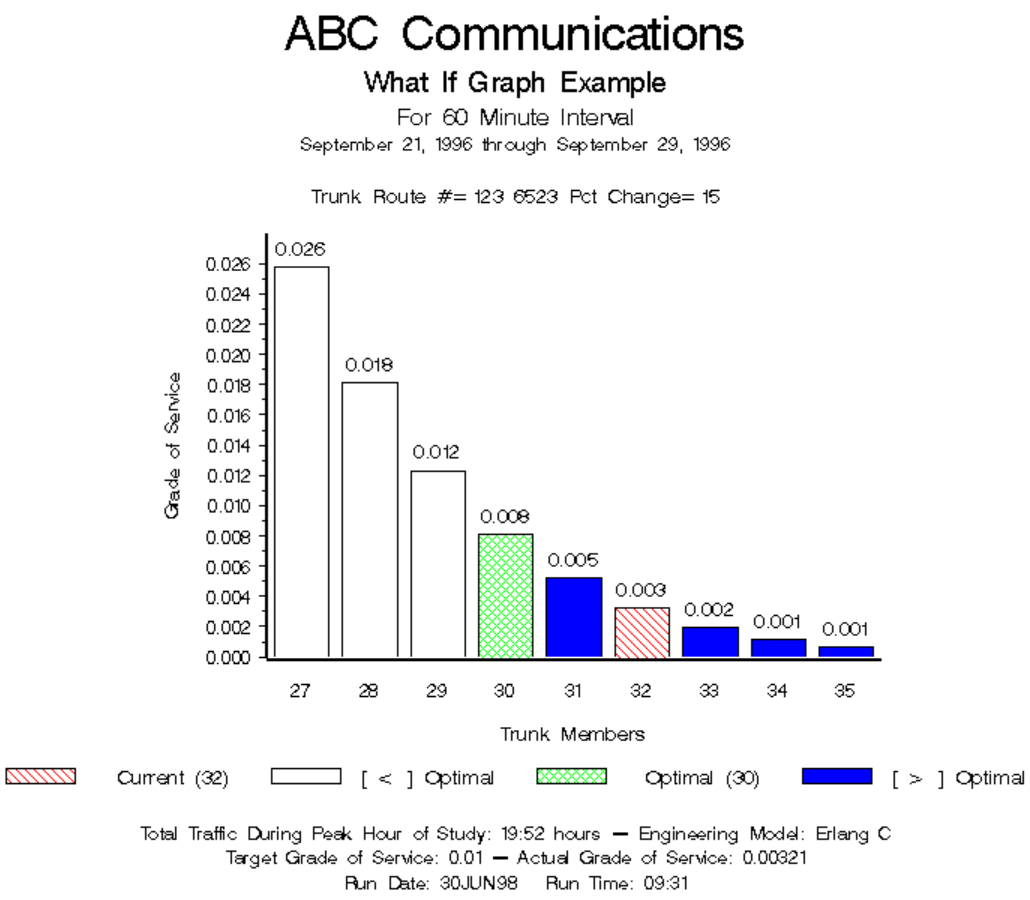
Date	Hour	Busy Hour	Busy Type	Over flows	Total Attempts	Total Usage (CCS)	Avg Hold Time (minutes)	% Blockage	Required Trunks
09/25/1996	2:00:00		BB	4	1,516	46	0.02	0.00	5
	7:00:00			3	1,671	31	0.01	0.00	4
	9:00:00			3	1,592	36	0.01	0.00	5
	10:00:00			4	4,073	45	0.01	0.00	5
	11:00:00			2	2,089	24	0.01	0.00	4
	17:00:00			345	999	8	0.00	0.35	3
Date Sum				**360**	**11,940**	**190**		**0.03**	**12**
Date Mean				**60**	**1,990**	**32**			**4**
09/26/1996	3:00:00		BB	1	3,003	19	0.00	0.00	4
	7:00:00			1	3,626	14	0.00	0.00	3
	10:00:00			0	4,220	10	0.00	0.00	3
	11:00:00			1	3,453	18	0.00	0.00	4
	15:00:00			0	4,795	12	0.00	0.00	3
	20:00:00			1	4,170	12	0.00	0.00	3
Date Sum				**5**	**23,266**	**85**		**0.00**	**7**
Date Mean				**1**	**3,878**	**14**			**3**
09/27/1996	4:00:00			0	5,201	5	0.00	0.00	2
	5:00:00		BB	7	6,717	139	0.15	0.00	10
	9:00:00			0	3,778	9	0.00	0.00	3
	11:00:00			0	4,194	7	0.00	0.00	3
	22:00:00			0	3,123	10	0.00	0.00	3
Date Sum				**8**	**23,012**	**170**		**0.00**	**11**
Date Mean				**2**	**4,602**	**34**			**4**
Trunk Group Sum				**373**	**58,218**	**445**		**0.01**	**21**
Trunk Group Mean				**22**	**3,425**	**26**			**4**

Report: SUGI2000 **Run Date and Time: 17JAN2000 20:01** **Page: 1**

TDC&E system, courtesy of Applied Digital Access, Inc.

SAMPLE REPORTS

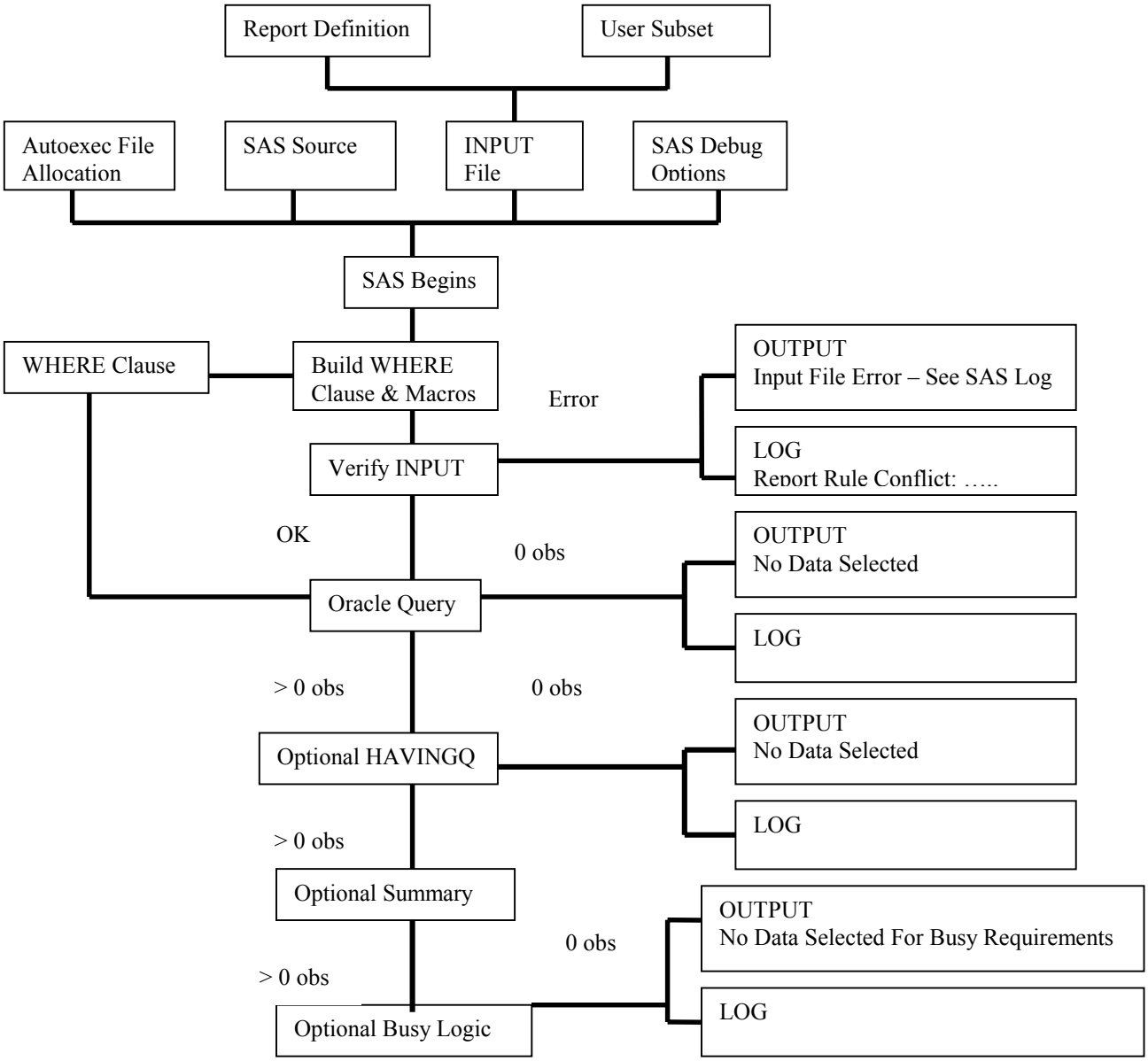
This is an example of output from the what if custom graph.



TDC&E system, courtesy of Applied Digital Access, Inc.

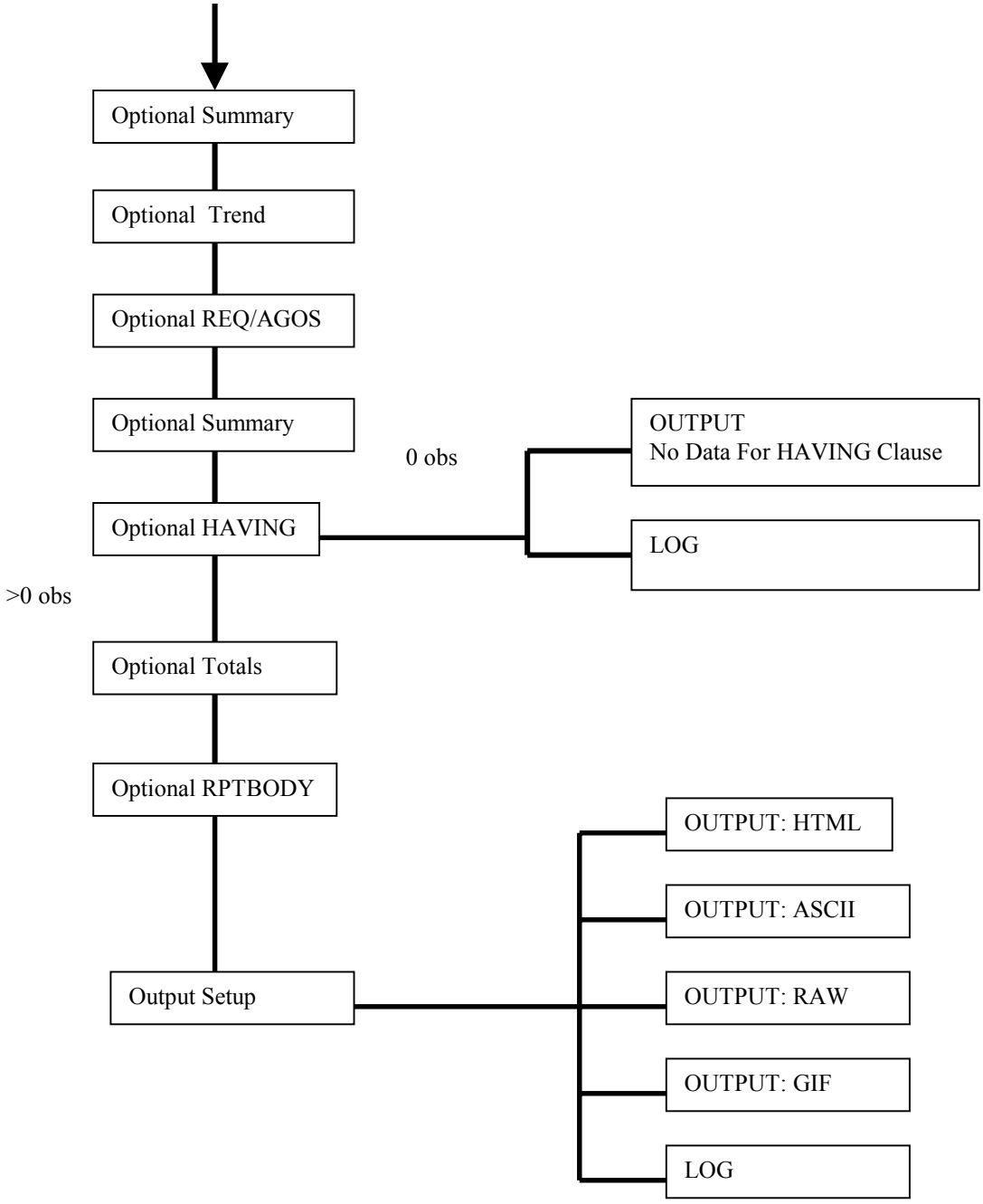
Report Generation SAS Source Code Flow Chart

The following flow chart illustrates the process of report generation. *Report Definition* and *User Subset* are defined in the GUI interface and then *Report Execution* begins, building the *INPUT file* from the *Report Definition* and *User Subset*. A call to SAS brings in an *AUTOEXEC* to allocate the appropriate files, a specific model or custom report (*SAS Source*), and *Debug Options* which control the SAS log contents. SAS reads the *INPUT file* and *Builds WHERE Clause & Macros*. The *Verify Input* insures that the INPUT file syntax and parameters relationships are correct before the *Query Oracle* begins. If the *INPUT file* does not pass the VERIFY criteria, SAS execution ends. A message is sent to the *OUTPUT, 'Input File Error - See SAS Log'* and a detailed error message is written to the log. If the *INPUT file* passes the *VERIFY Input*, the *Query Oracle* is executed. If the result of the query is 0 observations, a message is sent to the OUTPUT, 'No Data Selected' and processing does not continue. If the *Query Oracle* resulted in observations, the option *HAVINGQ* can be used to subset the data based on ASSIGN variables created immediately after the Oracle Query.. This is followed by an optional S*ummary* module which summarizes the data based on PROC SUMMARY and DATA step routines which determine FIRST/LAST values in time, MODE, and recalculation of ASSIGN variables. The Busy module identifies telecommunications busy days/time using bouncing busy, average busy, peak busy, time consistent busy, and predetermined busy.



TEC Associates Indianapolis, IN (317) 823-0642 diane_b@tecassociates.com

Another optional S*ummary* follows the Busy module, allowing user to summarize the results of the busy logic. *Trend* capabilities include linear trending, user specified growth factors in the form of incrments, percents, and actual an a combination of linear and growth trending. *REQ/AGOS* calculates teleommunications capacity using Erlang B, Erlang C, or Poisson formulas or engineering table lookups. Again, *Summary* follows the REQ logic in order to allow summarization of REQ calculations. An optional HAVING module allows subset after all processing is complete. If this results in 0 observations, output is generated with a message to that effect. Users can then specify report T*otals* which are very robust allowing multiple levels and multiple total lines per level. The *RPTBODY* routine allows the body of the report to be subset of eliminated, leaving the totals based on all data prior to rptbody logic. This is used for telecommunication thresholding. Finally, *OUTPUT* is generated in the form of HTML, text, and/or GIF.



TEC Associates Indianapolis, IN (317) 823-0642 diane_b@tecassociates.com

TABULAR MODEL: INPUT FILE

The following sample illustrates the INPUT file for the tabular model. This is the INPUT file which was used to produce the sample tabular report shown in the previous section. Note that the actual INPUT file allows only one parameter per line; an ASSIGN and RPTVAR shown in this section wrap only for purposes of display here. All parameter specifications include 3 values, separated by the tilde (~). A middle value of V indicates a holding position; the value is not used for the given parameter.

Parameters shown in the subset section are used to build the WHERE clause that will be used in the SQL query to Oracle. A middle parameter of L is list of values (SAS IN operator), R is range, I is like, and other valid values are standard such as EQ, LE, LT, GE, and GT. Parameters other than the subset parameters are converted to macro variables through many text transformations. These macro variables control the report layout, data summarization, sort order, totals, and new variables.

The primary macro variables for the report layout include ORDBY, BYVAR, REFVAR, IDVAR, ORDVAR, LABELS, and FORMATS. The ORDBY parameter controls the sort order on the report. Variables specified in the BYVAR form the BY line. REFVAR variables are variables associated with the BY line, such as address, and are shown on the line immediately following the BYVAR variables. IDVAR contains ID variables, which will be left most in the report body and consecutive like values will be blanked out. ORDVAR variables makeup the remainder of the variables and will follow the IDVAR variables in the body of the report. The LABEL and FORMAT parameters are used to build LABEL and FORMAT statements.

The section of busy parameters define the telecommunications workload busy logic to be performed. The assign section defines new variables to be created. The assign parameter allows the user to create new variables using any valid SAS statement. In addition, the ACCUM and ROLL parameters are used for accumulation and rolling accumulation of values of other variables in the report body. The ACRLINIT parameter controls when the accumulation statistics are initialized and the ROLLNBR specifies the number of rolling rows to include in the accumulation.

Data management parameters include GRPBY, RPTVAR, TOTBY, and TOTVAR. If a GRPBY parameter exists, the result of the Oracle query is summarized by the GRPBY and RPTVAR variables to build the report body. The GRPBY variables define the level of summarization and the RPTVAR variables define the statistics and the associated variables. The GRPBY parameter becomes the CLASS statement in PROC SUMMARY and the RPTVAR parameter is used to build the ID, VAR, and OUTPUT statements in PROC SUMMARY. The RPTVAR parameter can specify any valid PROC SUMMARY statistic and application specific statistics; FIRST, LAST, MODE, and ASSIGN. Assign indicates that the assign statement should be executed after summarization. The TOTBY and TOTVAR parameters are very similar to the GRPBY and RPTVAR parameters, except they are used to build the total lines rather than the report body. The syntax is identical. However, multiple TOTBY and TOTVAR parameters can be coded. Thus, a tabular report can have multiple levels of total, as well as multiple total lines per level.

The following WHERE clause is built from the subset section of the INPUT file:

```
WHERE
CO_NAME in('120 4623')
 AND CO_TYPE in('LOAD BAL ISLU')
 AND '25SEP96:00:00:00'dt <= DATE <=
'27SEP96:23:59:59'dt
 AND STUDY_NA in('rochny5e.mkt','rochny5e.trk')
;
```

TEC Associates Indianapolis, IN (317) 823-0642 diane_b@tecassociates.com

The following is the INPUT file for the tabular model and sample report shown:

```
# /ADA/TDCE/SASADA/INPUT/SUGI2000.TXT
TITLE1~T~SUGI TELECOM
TITLE2~T~TEC Associates
TITLE3~T~Indianapolis, Indiana
################## subset ##################
CO_TYPE~L~LOAD BAL ISLU
CO_NAME~L~120 4623
DATE~R~25SEP96:00:00:00
DATE~R~27SEP96:23:59:59
STUDY_NA~L~rochny5e.mkt
STUDY_NA~L~rochny5e.trk
SUSPECT~V~INCLUDE
################## options #################
IDBLANK~V~YES
CO_SORT~NOPARSE~C
INTERVAL~V~60
ORIENT~V~PORTRAIT
VIEW~V~BUSY
KEY~V~SUGI2000
LINES~V~48
ORIENT~V~LANDSCAPE
BYALIGN~V~LEFT
BYORIENT~V~VERTICAL
STYLE~V~PLAIN
STYLE~V~RAW
#################### busy #################
BUSYVAR~V~USAGE
BUSYCALC~V~BB
BUSYOUT~V~ALL
BUSYHRS~V~1
################# assign ##################
ASSIGN~V~HOLDTIM =
ROUND(USAGE/PEG_COUN*.6,.01)
ASSIGN~V~PERTBLK =
ROUND(USR1/PEG_COUN,.01)
ASSIGN~V~CHECK   =
TranWrd(BUSYID,"Busy","<CENTER><IMG
SRC=../TESTTEC/Check.gif HEIGHT=10
WIDTH=15></CENTER>")
############### report layout  ###############
GRPBY~V~RPT_NAME DAILY TIME
RPTVAR~V~LAST(ENG_METH CAP_ENG_ TYPE)
SUM(USR1 PEG_COUN USAGE) MEAN(HOLDTIM)
ASSIGN(PERTBLK CHECK)
ORDBY~V~RPT_NAME DAILY TIME
BYVAR~V~RPT_NAME
REFVAR~V~ENG_METH CAP_ENG_ TYPE GOS
IDVAR~V~DAILY
TOTBY~V~RPT_NAME DAILY
TOTBY~V~RPT_NAME
TOTVAR1~V~SUM(USR1 PEG_COUN USAGE)
ASSIGN(PERTBLK REQ)
TOTLAB1~V~Sum
TOTVAR2~V~MEAN(USR1 PEG_COUN USAGE
REQ)
TOTLAB2~V~Mean
ORDVAR~V~TIME CHECK BUSYLAB USR1
PEG_COUN USAGE HOLDTIM PERTBLK REQ
################ labels/formats ##############
CHECK~LABEL~Busy Hour
BUSYLAB~LABEL~Busy Type
RPT_NAME~LABEL~Trunk Group
ENG_METH~LABEL~Engineering Method
TYPE~LABEL~Engineering Type
DAILY~LABEL~Date
DAILY~FORMAT~MMDDYY10.
TIME~LABEL~Hour
TIME~FORMAT~TIME8.
USR1~LABEL~Overflows
USR1~FORMAT~COMMA10.
PEG_COUN~LABEL~Total Attempts
PEG_COUN~FORMAT~COMMA10.
USAGE~LABEL~Total Usage (CCS)
USAGE~FORMAT~COMMA10.
REQ~LABEL~Required Trunks
REQ~FORMAT~8.
GOS~LABEL~Grade of Service
HOLDTIM~LABEL~Avg Hold Time (minutes)
HOLDTIM~FORMAT~8.2
```

SELECTED SOURCE CODE

Partial source code for summarizing the body of the report and creating report totals is shown in the following sections.

SUMMARY Include

The SUMMARY include is called if the GRPBY parameter is specified in the INPUT file. The GRPBY macro variable becomes the CLASS statement. The RPTVAR parameter is parsed to create RVARI, RVARV, and RVARO. RVARI contains variables specified by the user with the TXT statistic and the RVARV macro variable contains all variables specified with a valid PROC SUMMARY statistic. The RVARO macro variable contains valid syntax for PROC SUMMARY and is built from the RPTVAR statistic and variables. After PROC SUMMARY is executed, any variables specified as ASSIGN in the RPTVAR parameter are built from the resultant SAS data set. Additional DATA step routines determines statistics, FIRST, LAST, and MODE. These are separate routines and the result is merged with the ORACLEIN data set created in the code below.

```
%macro summary;

  /* Find LAST statistic value */
 %if &rvarL ^= %str( ) %then %do;
       proc sort data = oraclein;
          by &ordby;
       run;
      data last;
        set oraclein;
        by &ordby;
       if last.&&grpbys&grpbysw;
       keep &grpbys &rvarL;
     run;
%end;

/* Summarize statistics available in PROC SUMMARY */
proc summary data = oraclein nway missing;
  class &grpbys;
  var &rvarv;
  output out = oraclein (drop = _type_
                rename=(_freq_=count))
   &rvaro
    ;
run;

/* Join LAST and PROC SUMMARY results */
%if &rvarL ^= %str( ) %then %do;
      data oraclein;
        merge oraclein last;
        by &grpbys;
      run;

   proc datasets library = work nolist;
      delete last;
   run;
%end;

/* Calculate ASSIGN variables */
%if &rvara ^= %str( ) %then %do;
      data oraclein;
        set oraclein;
       %if &asoran > 0 %then %do a=1 %to &asoran;
            %if  %parseit(%upcase(&rvara),
                  %upcase(&&avora&a))  > 0 or
                  %upcase(&&avora&a) = DO or
                  %upcase(&&avora&a) = END
                  %then %do;
                       %unquote(&&asora&a);
             %end;
       %end;
      run;
%end;

%mend summary;
```

TOTBY Include

The TOTBY include is called if the TOTBY parameter is specified or GRAND=YES on the INPUT file. The TOTBY macro variable becomes the CLASS statement. The TOTVAR parameter is parsed to create TVARI, TVARV, and TVARO, just as described above with RPTVAR. The primary difference between the SUMMARY include and TOTBY include is that multiple TOTBY and TOTVAR parameters can be specified, where only a single GRPBY and RPTVAR parameter can be specified. TOTBYN represents the number of TOTBY parameters and TOTVARN represents the number of TOTVAR parameters. PROC SUMMARY and the steps which follow are executed multiple times for each combination of TOTBY and TOTVAR. Not shown below are routines for the application specific statistics, FIRST, LAST, MODE, and ASSIGN. These are separate routines and the result merged with the ORACLEIN data set created in the code below.

```
%macro totby;

  %if &totbyn > 0 %then %do t=1 %to &totbyn;

      %if &tvarn > 0 %then %do l=1 %to &tvarn;

           proc summary data=oraclein nway;
             class &&totby&t;
             id &&tvari&l;
             var &&tvarv&l;
             output out = tot
              &&tvaro&l;
           run;


      /* Find LAST statistic value */
      %if &&tvarL&l ^= %str( ) %then %do;
```

```
        proc sort data = oraclein;
           by &ordby;
        run;
      data last;
        set oraclein;
        by &ordby;
        if last.&&ordby&ordbyw;
        keep &&totby&t &&tvarL&l;
      run;
      data tot;
        merge tot last;
        by &&totby&t;
      run;
      proc datasets library = work nolist;
        delete last;
      run;
  %end;

  data tot;
    set tot;
    totbyn  = &t;
    totvarn = &l;
    /* Re-ASSIGN on summary lines */
    %if &&tvara&l ^= %str( ) %then %do;
        %do rc=1 %to &&tvaraw&l;
            %if &assignn > 0 %then %do c=1 %to
                &assignn;
            %if &&tvara&l&rc = &&asvar&c or
                %upcase(&&asvar&c) = DO or
                %upcase(&&asvar&c) = END %then
                %do;
                %unquote(&&asgn&c);
            %end;
        %end;
      %end;
    %end;
  run;

  %if &t=1 and &l=1 %then %do;
     proc datasets library=work nolist;
       change tot = sassum;
     run;
  %end;
  %else %do;
     proc datasets library=work nolist;
       append base=sassum data=tot;
     run;
  %end;

 %end;

%end;
```

```
data oraclein;
  set oraclein (in=oin)
      sassum   (in=sin);
  if oin then do;
      totbyn = 0;
      totvarn = 0;
  end;
run;

%mend totby;
```

OUTSETUP Include

The OUTSETUP include contains many routines to prepare the final SAS data set for reporting. Code for this module will not be shown here. The following functions accomplished by the OUTSETUP include:

- Sorts report dataset:
  - user specified ORDBY
  - totbyn and totvarn internal variables
- Blanks out like consecutive values for ID variables
- Builds accumulation statistics
- Includes user specified labels and formats
- Builds labels for total lines
- Build HTML macro variables

HTMLOUT Include

The HTMLOUT include executes custom built stored and compiled macros that format the HTML output using macro variables built from the INPUT file parameters. Some of the application specific features of the custom HTML macro:

- User controlled placement of BY Line
- REFVAR for variables associated with BY variable
- Bold selected rows
- First page of same BY has all title lines; subsequent pages only title2 and BY line

CONCLUSION

The SAS component of the TDC&E system contains many custom routines specific to the telecommunications industry, but the approach and design of report generation is fundamentally the same across all industries. The INPUT file, the macro variables, the macros, and the include libraries are for the most part generic in nature. The TDC&E system has a strong foundation which can be expanded within the industry and as technology changes. Future capabilities include expanding the forecasting functionality and output formats.

ACKNOWLEDGEMENTS

Thanks to Applied Digital Access employees who I have worked with closely for almost 2 years on this project, including Dale Boyll, Jeff Evans, Allen Hackney, Stan Hawthorne, Guchi Jaurre, Dan McCann, Harry Meeker, Thomas Moore, Roger Sherfick, and Darlene Targett. Also thanks to Jeff Kubea, TEC Associates, who worked with me in writing the SAS component of this system, focusing on PROC TABULATE, all graphics output, and comma separated output.

For additional information, contact

Diane E. Brown
TEC Associates
8340 Galley Court
Indianapolis, IN 46236

(317) 823-7036
diane_b@tecassociates.com