ABSTRACT
LabOne has recently experienced unparalleled diversification and growth of its core businesses. This fact, along with significant changes in the diversity and magnitude of informational needs of LabOne Statistical Engineering clients, both internal and external, has created a paradigm shift in how we process, present, and distribute the requested information. We will discuss the evolution of our departmental goal to provide such value-added statistical and historical information, and the process changes we implemented to make our goal a manageable, auditable reality.

With the implementation of our corporate intranet came a change in our departmental strategy for delivering all this information. The following paper presents the strategy and the steps we took in meeting these objectives prior to our web enabling endeavors.

HISTORY
LabOne began as a laboratory providing health and lifestyle risk assessment testing to insurance underwriters, testing blood, urine or saliva specimens for the likelihood of nicotine addition, substance or alcohol abuse, or other chronic health problems. SAS® was first licensed at LabOne in 1990 as a means of producing statistical and historical reports for clients, and it quickly became apparent that the ability to provide such information to clients was going to be a key differentiator of LabOne over its competitors. Monthly and quarterly reports were implemented summarizing such information as test results as compared to national benchmarks, and laboratory turnaround time on specimen processing. As literally hundreds of clients would be receiving such reports, it became apparent that a robust production environment would be required so the generation of such reports would be efficient and timely.

ENTER STAGE LEFT: SAS/AF
The Statistical Engineering department quickly saw the need to turn over this ongoing reporting responsibility so that they could continue with the task of development to provide clients with other valuable information. The decision was made to turn this responsibility to the computer operations organization, whose primary function was to do regularly scheduled database moves and maintenance. But these computer operators had no knowledge of SAS. So we developed SAS CURE, a front end menu on our VMS operating system using SAS/AF, giving the operators access to the SAS programs required to run and produce these regularly scheduled reports.

A DATASET-DRIVEN SOLUTION
The first basic requirement of the solution was that it needed to be consistent with the current application. The number of computer operators was growing as the company grew, and employee turnover had become an issue. So we needed to keep it simple to operate, and ideally, the changes really needed to be transparent to the users. Also, the packing list that they used in facilitating distribution needed to be automated in order to keep it up to date without human intervention.

After several brainstorming sessions, we concluded that the most feasible solution was to modify the AF application to submit a batch job which was generated not from a series of SUBMIT commands, but instead, we generated the batch job from code that was feasible solution was to modify the AF application to submit a batch job which was generated not from a series of SUBMIT commands, but instead, we generated the batch job from code that was conditioned built by SAS/MACRO language, based on the values of parameters defined in a controlling dataset.

The SCL code was then stripped down to a basic SUBMIT block that executed a program called PRODSCL:

```
OPTIONS NOXWAIT;
%LET RUNDID=TEST;
%LET PRINLOG=Y;
%INCLUDE 'SAS_PGMS:[PROGRAMS]PRODSCL.SAS';
RUN;
```

NOXWAIT returns control to the SAS session automatically after the specified command is executed. This is necessary because of the CALL SYSTEM commands that we process to handle things such as email, file transfer, and printing in our production environment.

The parameters (macro variables) that are required are RUNDID, which identifies which set of reports to run (daily, weekly, monthly, post billing, and so forth). PRINLOG determines whether a packing list of the reports will be generated or not. PRODSCL in turn will build and execute a large batch jobstream based upon the records it identifies in a master dataset called STATLIST, which contains a record or records for each report in production (Appendix II shows a contents of STATLIST. Appendix III of this paper replicates PRODSCL.SAS in a very small font,
Due to proceeding space constraints. A more readable version will be available at my presentation or can be requested from me via e-mail or fax. See contact information below.)

A comment about the use of the option NOSYNTAXCHECK is in order here. By default, once SAS encounters an error, it will only perform syntax checks on subsequent statements. NOSYNTAXCHECK allows us to complete subsequent jobs in the job stream even if a previous job errors out (remember, the entire process is a bunch of potentially unrelated SAS programs executing back to back). This way, when we do have a problem in one program in the jobstream, the others will process normally and not require re-running. This is a very handy, but potentially dangerous, feature. If you choose to use it, do so wisely.

While a complete discussion of the processing that PRODSCL does is beyond the scope of this introductory paper, the following outline describes the basic processing that it does:

1. Create current beginning and ending dates for all possible time periods (today, previous day; last week, month, quarter, or year; etc.).
2. Create a current master table of all possible client company ID's for all 3 lines of business along with their company name.
3. Generate a list of the reports that will be run based upon the entries in dataset STATLIST that match the current RUNID.
4. If it has been requested, print the physical packing list for the operators to reference in distributing the upcoming reports as they generate.
5. Count the number of report combinations to be run.
6. For each combination, execute a macro which will:
   6.1. Create macro variables from all the necessary dataset elements: report name, frequency, recipient, number of copies, output type and device, and so forth.
   6.2. String multiple company ID's, test ID's, and state ID's together into one large macro variable so that the report need only be run once for any number of companies, tests, or states.
   6.3. PROC PRINTTO to separate LIS and LOG files so that reports can be printed separately and to facilitate troubleshooting problems with specific reports later.
   6.4. Finally, execute the SAS program.
   6.5. Execute system command(s) to distribute output according to the output device and location specified.
   6.6. Clean up work datasets and macro variables.
   6.7. Continue with next report.

"IT'S A FEATURE, NOT A BUG"

The dataset STATLIST began as a basic control dataset identifying report names, users, and output destinations. As enhancements to the system have been suggested and implemented, the program has become more complex, and the functionality has increased significantly. A few of the kinds of functionality we have been able to introduce at the request or suggestion of our internal and external customers are:

- The addition of color graphics output.
- Forced prioritization of job sequence. This allows us to schedule long running jobs at the end, or to arrange prerequisites to particular jobs.
- Generation of spreadsheet and other supported file formats.
- Automatic generation of emails (both internal and SMTP) containing either textual information or attachments.
- Generation of email notifications updating production report status for IT personnel or recipients.
- Generation of faxed reports to predetermined sites (in conjunction with a "user exit" to an internal application).
- Support for "special day processing"; i.e., determination of day of month and specific program execution based upon that.

FUTURE ENHANCEMENTS

The majority of the enhancements we have planned for this system are web-based and consequently covered in another paper (Paper Number p240-25, in the Posters section). However, there are features which are not necessarily web related that we have considered and hope to implement in the near future. We have automated the process of reporting both text and graphics to paper, email, fax, and, now, web; sufficient output types and delivery methods seem to exist for the near term. Therefore, the scope of the enhancements we are pursuing are maintenance related.

It has become increasingly difficult to proactively monitor the ongoing integrity of each program and its underlying data. We hope to automate the process of identifying such problems as they run, rather than having the recipient of the information alert us to such problems. For example, we hope to produce automated emails that consolidate log error details from the numerous reports that are running. Other means of quickly identifying data inconsistencies are similarly in the works.

CONCLUSION

The intent of the preceding paper has been to describe how LabOne's Statistical Engineering Department has handled the growing need for information by its internal and external customers while keeping the administrative overhead involved to a minimum. Our basic methodology has been presented, along with some ideas for implementation and enhancement. While some code specific to our systems and processes exist, a similar approach could certainly be taken by other organizations trying to accomplish the same thing.

CONTACT INFORMATION

If you have any questions, comments or feedback, please do not hesitate to contact the author at:

Steven Beakley
LabOne, Inc.
10101 Renner Boulevard
Lenexa, Kansas 66219
Work Phone: (913) 577-1380
Fax: (913) 888-4160
Email: steve.beakley@labone.com
Web: http://www.labone.com
DATA Warehousing

INIT:
* CALCULATE DATES TO BE DISPLAYED ON INITIAL SCREEN;
  CUR=DATE();
  D=DAY(CUR);
  NEWCUR=CUR-D;
  Y=YEAR(NEWCUR);
  M=MONTH(NEWCUR);
  BDATE=PUT(Y,4.)||PUT(NEWCUR,MMDDYY4.);
  IF M <10 THEN DO;
    BDATE=TRIM(Y)||'0'||TRIM(LEFT(M))||'01';
  END;
  ELSE DO;
    BDATE=TRIM(Y)||TRIM(LEFT(M))||'01';
  END;

  return;

MAIN:
* PERFORM SCREEN INPUT VALIDATION CHECK;
  IF BDATE > EDATE THEN DO;
    ERROROFF BDATE EDATE;
    ALARM BDATE=TRIM(Y)||TRIM(LEFT(M))||'01';
    END;
  ERRORON BDATE EDATE;
  return;

TERM:
  IF _STATUS_ = 'C' THEN RETURN;

SUBMIT:
* MONTHLY KIT VALIDATION FOR U.S. INSURERS;
  ASSIGN VARIOUS MACRO VARIABLE INPUTS;
  %LET BDATE=*&BDATE;
  %LET EDATE=*&EDATE;
  %LET LEGAL=1;
  %LET EMTH=.
  %LET BMTH=.
  %LET RSORT=1;
  %LET FREQ="M";
  %LET FREQ="Q";
  %LET EMTH=.
  %LET BMTH=.
  %LET RSORT=1;
  %LET FREQ="Q";
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  RETURN;

SUBMIT:
* QUARTERLY KIT VALIDATION FOR U.S. CLIENTS (AFTER REASSIGNING BEGIN AND END DATES);
  %LET BDATE=*&BDATE;
  %LET EDATE=*&EDATE;
  %LET LEGAL=1;
  %LET EMTH=.
  %LET BMTH=.
  %LET RSORT=1;
  %LET FREQ="Q";
  ASSIGN OUTPUT LISTING TEXT FILE NAME;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  RETURN;

SUBMIT:
* QUARTERLY KIT VALIDATION FOR U.S. EXAMINERS;
  %LET RSORT=2;
  ASSIGN OUTPUT LISTING TEXT FILE NAME;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  %INCLUDE 'SAS_PGMS:[PROGRAMS.HIST]AKITVAL.SAS'/SOURCE2;
  RETURN;

SUBMIT:
* AGAIN, THE PROCESS WOULD CONTINUE FOR CANADIAN DIVISION REPORTS...
  ALL THE QUARTERLY REPORTS ARE BEING PRINTED TOGETHER TO ELIMINATE OTHER REPORTS BEING PRINTED BETWEEN THESE;

DATA_NULL:
  CALL SYSTEM ('PRINT AKITVMCL.LIS/QUE=LCONF_HPS0000/FORM-HPKITVAL');
  CALL SYSTEM ('PRINT AKITVMEL.LIS/QUE=LCONF_HPS0000/FORM-HPKITVAL');
  CALL SYSTEM ('PRINT AKITVMCL.LIS/QUE=LCONF_HPS0000/FORM-HPKITVAL');
  CALL SYSTEM ('PRINT AKITVMEL.LIS/QUE=LCONF_HPS0000/FORM-HPKITVAL');

ENDSUBMIT:
END;
RETURN;

Appendix I: Pre-Historic SCL Entry
Data Set Name: DATA.STATLIST                           Observations:         2888
Member Type:   DATA                                    Variables:            32
Engine:        REMOTE6                                 Indexes:              0
Created:       15:46 Monday, January 10, 2000          Observation Length:   433
Last Modified: 9:01 Friday, January 21, 2000           Deleted Observations: 1
Protection:                                            Compressed:           NO
Data Set Type:                                         Sorted:               NO
Label:

-----Engine/Host Dependent Information-----
Data Set Page Size:       32768
Number of Data Set Pages: 39
File Format:              607
First Data Page:          1
Max Obs per Page:         75
Obs in First Data Page:   65
Filename:                 SAS$DRA2:[SAS_DATA]STATLIST.SASEB$DATA
Host Format:              AXP
Disk Blocks Allocated:    2502

-----Alphabetic List of Variables and Attributes-----

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Pos</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BUSID</td>
<td>Char</td>
<td>3</td>
<td>0</td>
<td>BUSINESS SERVICE ID</td>
</tr>
<tr>
<td>15</td>
<td>COMPID</td>
<td>Char</td>
<td>4</td>
<td>365</td>
<td>COMPANY ID</td>
</tr>
<tr>
<td>18</td>
<td>COPIES</td>
<td>Num</td>
<td>8</td>
<td>371</td>
<td>NUMBER OF COPIES TO BE PRINTED</td>
</tr>
<tr>
<td>3</td>
<td>COPYTO</td>
<td>Char</td>
<td>100</td>
<td>19</td>
<td>EMAIL CARBON COPY</td>
</tr>
<tr>
<td>21</td>
<td>CPOPT</td>
<td>Char</td>
<td>1</td>
<td>383</td>
<td>COLOR PRINTER OPTION</td>
</tr>
<tr>
<td>9</td>
<td>DIR</td>
<td>Char</td>
<td>8</td>
<td>244</td>
<td>PROGRAM DIRECTORY</td>
</tr>
<tr>
<td>19</td>
<td>DSET</td>
<td>Char</td>
<td>2</td>
<td>379</td>
<td>DATASET INDICATOR</td>
</tr>
<tr>
<td>17</td>
<td>EXECREC</td>
<td>Char</td>
<td>1</td>
<td>370</td>
<td>EXECUTABLE RECORD INDICATOR</td>
</tr>
<tr>
<td>20</td>
<td>FACIL</td>
<td>Char</td>
<td>2</td>
<td>381</td>
<td>FACILITY</td>
</tr>
<tr>
<td>28</td>
<td>FILETYPE</td>
<td>Char</td>
<td>3</td>
<td>411</td>
<td>ATTACHMENT FILE TYPE</td>
</tr>
<tr>
<td>2</td>
<td>FORM</td>
<td>Char</td>
<td>16</td>
<td>3</td>
<td>PAPER FORM</td>
</tr>
<tr>
<td>16</td>
<td>FREQ</td>
<td>Char</td>
<td>1</td>
<td>369</td>
<td>REPORT FREQUENCY</td>
</tr>
<tr>
<td>4</td>
<td>GPHPRNTR</td>
<td>Char</td>
<td>20</td>
<td>119</td>
<td>GRAPHICS PRINTER</td>
</tr>
<tr>
<td>24</td>
<td>INHSE</td>
<td>Char</td>
<td>1</td>
<td>396</td>
<td>INHOUSE REPORT INDICATOR</td>
</tr>
<tr>
<td>5</td>
<td>MESSAGE1</td>
<td>Char</td>
<td>80</td>
<td>139</td>
<td>EMAIL MESSAGE LINE 1</td>
</tr>
<tr>
<td>29</td>
<td>NOTIFY</td>
<td>Char</td>
<td>1</td>
<td>414</td>
<td>EMAIL NOTIFICATION FLAG</td>
</tr>
<tr>
<td>23</td>
<td>PATH</td>
<td>Char</td>
<td>8</td>
<td>388</td>
<td>MENU PATH</td>
</tr>
<tr>
<td>12</td>
<td>PRINDEST</td>
<td>Char</td>
<td>20</td>
<td>277</td>
<td>OUTPUT PRINT DESTINATION</td>
</tr>
<tr>
<td>26</td>
<td>PRINTER</td>
<td>Char</td>
<td>8</td>
<td>400</td>
<td>OUTPUT PRINTER IDENTIFIER</td>
</tr>
<tr>
<td>27</td>
<td>PRIORITY</td>
<td>Char</td>
<td>3</td>
<td>408</td>
<td>PRIORITY INDICATOR</td>
</tr>
<tr>
<td>25</td>
<td>PROFILE</td>
<td>Char</td>
<td>3</td>
<td>397</td>
<td>PROFILE</td>
</tr>
<tr>
<td>13</td>
<td>RECIP</td>
<td>Char</td>
<td>60</td>
<td>297</td>
<td>REPORT RECIPIENT</td>
</tr>
<tr>
<td>8</td>
<td>REG</td>
<td>Char</td>
<td>2</td>
<td>242</td>
<td>REGION CODE</td>
</tr>
<tr>
<td>7</td>
<td>REM</td>
<td>Char</td>
<td>3</td>
<td>239</td>
<td>REMARK</td>
</tr>
<tr>
<td>10</td>
<td>REPNAMES</td>
<td>Char</td>
<td>20</td>
<td>252</td>
<td>REPORT NAME</td>
</tr>
<tr>
<td>11</td>
<td>REPORTS</td>
<td>Char</td>
<td>5</td>
<td>272</td>
<td>REPORT PARAMETER</td>
</tr>
<tr>
<td>30</td>
<td>RUNDAY</td>
<td>Num</td>
<td>8</td>
<td>415</td>
<td>RUN DAY FOR SPECIAL PROCESSING</td>
</tr>
<tr>
<td>14</td>
<td>RUNID</td>
<td>Char</td>
<td>8</td>
<td>357</td>
<td>RUN IDENTIFIER</td>
</tr>
<tr>
<td>32</td>
<td>STATE</td>
<td>Char</td>
<td>2</td>
<td>431</td>
<td>STATES TO BE PROCESSED</td>
</tr>
<tr>
<td>22</td>
<td>TEST</td>
<td>Char</td>
<td>4</td>
<td>384</td>
<td>TESTS TO BE PROCESSED</td>
</tr>
<tr>
<td>6</td>
<td>TYPE</td>
<td>Char</td>
<td>20</td>
<td>219</td>
<td>TYPE PARAMETER</td>
</tr>
</tbody>
</table>

Appendix II: STATLIST Contents
**SUMMARY:**

- This script defines several procedures for managing database connections and reporting.
- It uses SQL to select distinct company IDs and related data for various reports.
- The script is designed to run at specific intervals and can be customized for different needs.

**PROCEDURE:**

1. **PROCEDURE:**
   - Defines a company name and ID for an all rename procedure.
   - Sets up data blocks and connections to an Oracle database.
   - Populates a table with data from the database.
   - Creates a table for process logs.
   - Processes the data and generates reports.
2. **PROCEDURE:**
   - Selects distinct company IDs and processes data for specific reports.
   - Handles special cases and outputs reports.
3. **PROCEDURE:**
   - Selects the reports for the specified run ID and processes data accordingly.
   - Generates process logs and outputs reports.

**DATA:**

- Represents various data fields and connection parameters.
- Includes SQL queries for selecting and manipulating data.

**MACROS:**

- Defines macros for handling special cases and outputting reports.
- Manages the execution and priority of processes.

**OPTIONS:**

- Specifies line size and other printing options.

**PROGRAM:**

- Contains the main program logic and flow.
- Uses loops and conditional statements to process data and output reports.

**OUTPUT:**

- Outputs reports in a specific format.
- Processes logs and outputs reports.

**PROCESS LOG:**

- Contains a log of processes and outputs.
- Tracks the status and priority of processes.

**REPORTS:**

- Generates reports based on selected criteria.
- Includes parameters for outputting reports in different formats.

**TABLES:**

- Defines tables for storing data and process logs.
- Contains columns for storing specific data fields.

**SQL QUERIES:**

- Selects distinct company IDs and processes data for specific reports.
- Generates process logs and outputs reports.

**SQL FIELDS:**

- Defines fields in SQL queries for selecting and manipulating data.

**Variables:**

- Represents various data fields and parameters used in the script.
- Includes variables for handling special cases and outputting reports.