

Enabling Transitions of Care with SAS®

Chris Schacherer, Clinical Data Management Systems, LLC

ABSTRACT

As changes in healthcare payment models continue to evolve, provider organizations face a constant need for a strong data management and analytics capability to help them respond to shifting measurement and analysis requirements. Population Health Management (PHM) software and Business Intelligence (BI) solutions provide many of the required capabilities to identify gaps in care and measure clinical and financial performance. However, even with industry-leading packaged software solutions in place, custom data management solutions are often needed to respond quickly to specific requirements. SAS® software provides a wide array of functionality to fill these gaps. The current work describes how the FILENAME statement, PROC SQL, and DATA step were used to help providers in one accountable care organization (ACO) quickly identify when their patients were discharged from area hospitals and emergency departments so they could provide appropriate follow-up care.

INTRODUCTION

As the business of healthcare continues its transformation from payment for volume to payment for value, the definition of “good healthcare” has increasingly focused on making sure patients get the right treatment at the right time. This focus takes many forms—from patient centered medical homes and medication therapy management benefit programs to identification of fraud, waste, and abuse. One thing that all of these types of initiatives have in common is that they are dependent on data analysis to identify cohorts of patients, create metrics reflecting performance on some behavior (by the provider, payer, or patient), and provide targeted, actionable information to the people operating within the healthcare system to guide them in the behaviors that will lead to a more efficient and effective healthcare system. One example of such a data-driven process is the management of “transitions of care”. As described by the National Transitions of Care Coalition (2017):

Transitions of care are a set of actions designed to ensure coordination and continuity. They should be based on a comprehensive care plan and the availability of well-trained practitioners who have current information about the patient's treatment goals, preferences, and health or clinical status. They include logistical arrangements and education of patient and family, as well as coordination among the health professionals involved in the transition.

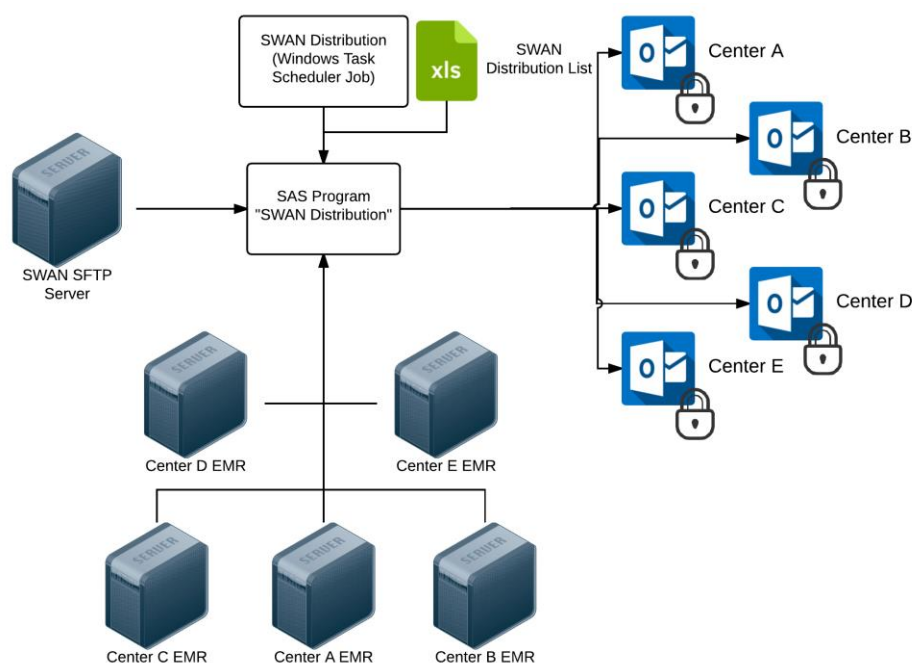
Specifically, the current work focuses on how SAS was used to assist in two specific types of transitions—those involving discharge from an In-Patient (IP) hospital stay and those involving discharge from the Emergency Department (ED). In both of these transitions, it is important that follow-up care with one's primary care provider is carried out where appropriate. However, it is often the case that hospital/ED patients receive their primary care from a physician who is not affiliated with the hospital where they got their In-Patient/ED care. Without a means of notifying primary care providers of discharges from these settings, the patient's care may be compromised and the patient may end up back in the hospital or emergency department. This scenario is both bad for the patient and bad for the payer (which is why additional focus is also placed on measurement of “readmission rates”).

STATEWIDE ALERT NOTIFICATION SYSTEM (SWAN)

In Iowa, transitions of care following IP/ED discharges have been one of the areas of focus for the Iowa Health Information Network (IHIN), the statewide Health Information Exchange. The StateWide Alert Notification System (SWAN) was implemented by IHIN in 2015. Within this system, hospitals voluntarily provide Admission, Discharge, & Transfer (ADT) records associated with Iowa Medicaid Enterprise members to a central repository. Medicaid Accountable Care Organizations (ACOs), similarly, provide a roster of their current, attributed patients to the repository. On a daily basis, the hospital/ED Admits and Discharges are fed to the central repository via Secure File Transfer Protocol (SFTP), the list of ADT records is matched to each ACO's patient list, and a list of patients admitted or discharged from an IP stay or ED visit is produced for the ACO. Each ACO can then download their daily list of ADTs and use that list to drive care-coordination activities—identifying admits and discharges associated with certain diagnoses (e.g., “sprained ankle” vs. “myocardial infarction”) and responding, as appropriate, with a call to the patient to follow-up with a primary care visit.

Clinical Data Management Systems, LLC worked with one of the Medicaid ACOs in the state to enable the member

healthcare organizations—stand-alone community health centers—to take advantage of these data and enable them to provide more consistent care transitions. As depicted in the following flow-chart, the solution described in the rest of the paper involves a SAS program that was scheduled with Windows Task Scheduler® to connect to the SWAN SFTP server (using the individual ACO's dedicated service account) every morning at 7:00 a.m. The program then processes the current day's report of all ADTs reported in the last 24 hours. Importantly, this file is parsed to identify the Iowa Medicaid members whose care has been attributed to each of the ACO health centers so that each center only receives ADT records associated with members under their care. The "SWAN Distribution" program also connects to the electronic medical record system at each of the member centers and extracts current data related to each patient so that when the care management team at the centers receive their morning ADT report they have localized information that will make the report more actionable (e.g., the patient's primary care provider, the date of their last clinic visit, contact information, etc.). These data are then put together into a report for each center and the care coordination team at each center (identified by the "SWAN Distribution List") receives this report via encrypted email a few minutes after the file is downloaded.



MACRO TO CONNECT SAS TO EMRS

One of the keys to all of the healthcare analytics programs written in SAS for the ACO is a macro that makes a read-only connection to all of the associated EMRs. It's important to note that many of today's healthcare organizational partnerships (ACOs, but also independent practice associations, IPAs) involve multiple stand-alone healthcare organizations that may well be utilizing disparate electronic health record and practice management systems. Therefore, any network-wide analysis will require making connections to multiple EMRs (and practice management, dental record, billing, and population health management systems). Instead of coding these connections into every analytical and data management program written, one macro was created to make a read-only connection to each of these systems using read-only service accounts to connect to each system's database.

The macro "**connect_all_dbs**" is used to maintain connections for the service accounts in each of the center EMRs and is stored in the "mainfile" macro library. As "connect_all_dbs" begins, the password associated with the service account for Center A's Microsoft SQL Server® database is read from a SAS dataset "svc" that is maintained on encrypted drive space to which only the service account (ADTsvc) has access. The password is read into the macro variable "emr_pass_center_a". The service password is then used to define a LIBNAME representing a database connection (in this case to the "DBO" schema on the "BigEMRCo" instance, using the SAS/ACCESS OLE connector. For more information on making database connections using SAS/ACCESS products, see Schacherer (2011), Schacherer & Detry (2010).

```

libname mainfile '\\hca1\.SAS Macros';
options mstored sasmstore=mainfile;

%macro connect_all_dbs / store source;

PROC SQL NOPRINT;
  SELECT readonlypw INTO :emr_pass_center_a
    FROM mainfile.svc
    WHERE center='center_a' AND
      system = 'EMR';
QUIT;
PROC SQL NOPRINT;
  SELECT readonlypw into :emr_pass_center_b
    FROM mainfile.svc
    WHERE center='center_b' AND
      system = 'EMR';
QUIT;
...

LIBNAME center_a OLEDB PROVIDER=SQLOLEDB OLEDB_SERVICES=NO
ACCESS=readonly/*REQUIRED=Yes*/
USER = "ADTsvc"
password=&dbpass
DATASOURCE = "192.168.100.101"
PROPERTIES = ('initial catalog'=BigEMRCo
              'Persist Security Info'=True)
SCHEMA = 'DBO';

LIBNAME center_a OLEDB PROVIDER=SQLOLEDB OLEDB_SERVICES=NO
ACCESS=readonly/*REQUIRED=Yes*/
USER = "ADTsvc"
password=&dbpass
DATASOURCE = "192.168.100.102"
PROPERTIES = ('initial catalog'=LittleEMRCo
              'Persist Security Info'=True)
SCHEMA = 'EMR';
...

```

SWAN DISTRIBUTION PROGRAM

Logging. Because the SWAN distribution program is developed to run unattended as a scheduled job every morning, the first step in the program is to establish a log for each execution of the file. In the event of a job failure (and, to enable more complete audit log of who touched which data, when), this log can be reviewed to give some insight into the source of the failure. As shown below, a macro variable, "log_file_path", is generated to provide a filename of the form "<job name>_<datetime of execution>". For example, when this job executed on September 9th, 2017, the log file was written to "SASLOG_SWAN_Distribution__2017-09-09_7.45.01.txt".

```

%LET logpath = \\hca1\SWAN\Logs;
%LET etl_program = SWAN_Distribution_;

%LET datestamped_log =&etl_program._%SYSFUNC(PUTN(%SYSFUNC(DATE()),
  YYMMDD10.))_%SYSFUNC(TRANSLATE(%SYSFUNC(
  PUTN(%SYSFUNC(TIME()), HHMM8.2)),.,:));

%PUT &datestamped_log;

```

```

DATA _NULL_;
CALL SYMPUT('log_file_path', '"' || RESOLVE('&logpath') || "\"SASLOG_" ||
RESOLVE('&datestamped_log') || ".txt"');
RUN;

%PUT &log_file_path;

```

A PROC PRINTTO step is then executed to redirect the output of the program's execution to the file identified by "&log_file_path". Note, that in this PROC PRINTTO step the "NEW" option is used to create a new log with each execution. An alternative approach to logging could be utilized to recycle the logs every week, every 30 days, etc. and spool the PROC PRINTTO redirect to an existing log.

```

PROC PRINTTO LOG=&datestamped_log NEW;
RUN;

```

With the logging data output to an external, persistent file, the SAS option to abend the execution of the program in response to any error (errorcheck=strict) is set such that SAS will immediately quit in response to any ERROR event—through WARNINGS and NOTES will not cause an abend.¹²

```

OPTIONS ERRORCHECK=STRICT ERRORABEND;

```

Database Connections & Other Libraries. With error logging and error processing complete, the "connect_all_dbs" macro is executed to give the program access to each Center's EMR and Practice Management databases.

```

libname mainfile '\\hcal\SAS Macros';
options mstored sasmstore=mainfile;

%connect_all_dbs;

```

Distribution List. Next, a PROC IMPORT step is executed to bring in the list of e-mail addresses for all recipients of the ADT reports. This list will be used later when each Center's individualized report is generated and emailed to the recipient list for that center.

```

PROC IMPORT OUT= swan_distrib_list
            DATAFILE= '\\hcal\SWAN\Distribution List.xlsx'
            DBMS=EXCELCS REPLACE;
            RANGE="Sheet1$";
            SCANTEXT=YES;
            USEDATE=YES;
            SCANTIME=YES;
RUN;

```

It should be noted that this list is maintained outside of the SWAN Distribution program, and although in this case it is kept in an Excel file, it could be generated from a human resources or practice management system to send the list to anyone with an appropriate title (e.g., "Care Management Team Lead").

¹ A conservative approach to error handling is taken in this case because, later in the program, emails with attachments containing personal health information (PHI) are sent to care managers at multiple clinics. Therefore, if an error (say, in a DATA step creating a new dataset) fails and an old dataset is not over-written, you would want to make sure that the program fails, rather than send patient data to the wrong clinic.

² The reader should note that when developing and testing programs that use this approach to error processing, forgetting to comment out this OPTIONS statement when you are developing, testing, and debugging will lead to significant frustration. When developing and debugging, do not forget to comment out this statement.

	A	B	C	D	E
1	center	email	Active	Start	End
2	CenterA	msmith@centera.org	Y	7/11/2017	
3	CenterB	tjones@centerb.org	Y	7/11/2017	
4	CenterA	jrollins@centera.org	Y	8/10/2017	
5	CenterC	ruth.watkins@centerc.org	Y	2/16/2017	
6	CenterC	john.thomas@centerc.org	N	3/17/2017	3/27/2017

Finally, a DATA Step is executed with a CARDS statement to provide the ability to use different naming conventions when referring to the different centers. From the “full_names” dataset, one can link the center’s “nickname” to “name_location”, for example.

```
DATA work.full_names;
INPUT nickname $ 1-9 fullname $ 10-30 abbreviation $ 31-40;
CARDS;
CENTERA Healthcare Center A Center A
CENTERB Healthcare Center B Center B
CENTERC Healthcare Center C Center C
CENTERD Healthcare Center D Center D
CENTERE Healthcare Center E Center E
;
RUN;
```

Identify & Download Today’s SWAN ADT File. At this point the program changes focus to identify and download the file of today’s ADT records from the SWAN System. As described by Schacherer (2012), the FILENAME statement is a powerful tool for interacting with the world outside of SAS. In this example, the program will be used to perform two different processing steps. First, the SWAN outbound directory accessible by the current user credentials will be checked for the existence of today’s file. Then today’s file will be downloaded and parsed into a SAS dataset.

In this first DATA step, the program needs to create a macro variable “swan_file_shortcode” to specify the root of the daily file’s name. Although the files are named and date-stamped for each day they are generated, the time at which they are generated can vary slightly. For example, the file for September 9, 2017 could be generated as “iowa_health_ADTs_20170909000001” (one second after midnight) or “iowa_health_ADTs_20170909000002” (two seconds after midnight). Therefore, the following DATA step creates the macro variable “swan_file_shortcode” to identify the appropriate filename with a wildcard (e.g., iowa_health_ADTs_20170909*.csv).

```
DATA _NULL_;
CALL SYMPUT('swan_file_shortcode',
            "'iowa_health_ADTs_'||
            compress(translate(put(date()+1, yymmdd10.), "", "-"))||
            '*.csv'");
RUN;

%PUT &swan_file_shortcode;
```

This wildcard filename will then be used to create the filename “swan” that uses the “SFTP” access method. As described previously (Schacherer, 2012), the filename acts as a reference to the access method, such that when the “swan” filename is referenced in the later DATA step to create the “dayfile” dataset, the INFILE statement that references it will, effectively, be executing the referenced access method defined in the FILENAME statement. In this case, a connection is made to the SWAN SFTP server using the specified credentials, changing the directory to “outbound” (where the daily files are dropped) and then issuing a listing command to list (and download to the DATA step) the names of any ADT files that match the wildcarded filename. Therefore, regardless of whether the file name

is 20170909000001, 20170909000002, or even 20170909083422, the name of the file for today's ADTs will be correctly identified.

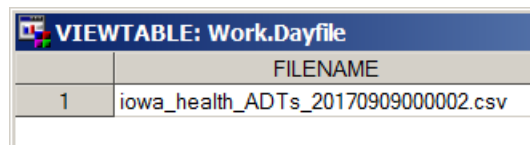
Please note: Unlike use of the FTP access method, use of the SFTP access method requires a secure shell (SSH) client to be installed on the machine where SAS is run. According to SAS (2008):

During execution of the SFTP access method in a SAS® FILENAME statement, the method dynamically launches an sftp or a psftp executable file. Launching this executable initiates an SSH client process that creates a secure connection to an OpenSSH server. All conversations across this connection are encrypted—from the user authentication to the data transfers. However, before you can use the SFTP access method in a SAS FILENAME statement, you must set up the SSH software that enables SAS to access a file on an OpenSSH server.

```
FILENAME swan SFTP options="-pw mysecretpassword"
USER='<myusername>'
HOST='sftp.secretsight.schacherer.com'
CD="outbound"
RECFM=F
LS LSFILE=&swan_file_shortcode;

DATA dayfile;
  INFILE swan;
  INPUT filename :$50.;
RUN;
```

Looking at the value of "filename" in the dayfile, you can see that in this particular case the filename is "iowa_health_ADts_20170909000002.csv".



VIEWTABLE: Work.Dayfile	
	FILENAME
1	iowa_health_ADts_20170909000002.csv

After generating this data point, the program continues to validate that a file for today was identified, and that there is only one file. This is achieved by using PROC SQL to select a count of the files identified in work.dayfile into the macro variable "file_exists". &file_exists is then evaluated to confirm that one and only one file of today's ADTs was identified in the outbound SFTP directory. As described elsewhere (Schacherer, 2011), a DATA _NULL_ step is used to evaluate the value of &file_exists. If the value is "1" (indicating a single file exists for today's processing), then the value of "terminator" is assigned a null string. If the value is any other number, the value of "terminator" is set to "ENDSAS;". Following the DATA _NULL_ step, the value of &terminator is resolved and either generates white space—ignored by SAS processing—or issues the command ENDSAS—ending the SAS session immediately (in which case the log file can be referenced to determine the form of the error).

```
proc sql noprint;
  select count(*) into :file_exists
  from work.dayfile;
quit;

DATA _NULL_;
  IF &file_exists = 1 THEN
    CALL SYMPUT('terminator','');
  ELSE
    CALL SYMPUT('terminator','ENDSAS;');
RUN;

&terminator
```

Assuming that SAS continues beyond this point, the name of today's file is selected into the macro variable "download_file", and that specific file is downloaded from the SFTP directory using another FILENAME statement and associated DATA Step. Unlike the previous FILENAME statement, the FILENAME statement for "swanfile" is referencing a specific file to be downloaded and does not issue an LS command. The program is now focused on downloading the specific file for today's processing. The DATA step references this file in the INFILE statement—parsing the .csv file with the appropriate delimiter (DLM=',') and handling delimiters within a value by recognizing those quoted strings (DSD). The INPUT statement then names and formats the data according to the SWAN data dictionary.

```
PROC SQL NOPRINT;
  SELECT filename INTO :download_file
  FROM work.dayfile;
QUIT;

FILENAME swanfile SFTP "&download_file" OPTIONS="-pw &mysecretpassword"
USER='myusername'
HOST='sftp.secretsight.schacherer.com'
CD="outbound";

DATA todays_swan_file ;
  INFILE swanfile FIRSTOBS=2 DLM=',' dsd;
  INPUT Medicaid_ID :$8. Location :$50. Admit_Date :$10.
        Patient_Lname :$25. Patient_Fname :$25. Patient_Type :$3.
        Medical_Service :$1. Discharge_Date :$10. ACO_Center :$20.
        Discharge_Disposition :$25. Patient_Account_Number :$20.
        Admit_DX_Code :$10. Admit_DX_description :$50.;

RUN;
```

The program has now produced a SAS dataset of today's ADTs (with the Medicaid ID, Patient Name, Admit and Discharge dates, as well as the admitting diagnoses and the ACO Center to which the member has been attributed for primary care). With these data, the program can now start to generate, for each primary care center, the list of inpatient and ED discharges on which they need to follow-up for any need care transition services.

SWAN MAIL Macro. Because there are multiple centers for which the same steps need to be executed, a macro is created. This macro, "swan_mail", will dynamically create the report for each center (where the center nickname is passed as a parameter to the macro), identify the report recipients associated with that center, and send an encrypted email (with the report attached) to the identified recipient list.

```
%macro swan_mail (center=);
```

After declaring the macro and the input parameter (center), a list of ADT records attributed to the center will be created using a PROC SQL step. For example, if the center name passed to the macro call was "CENTERA", the dataset "SWAN_CENTERA_01" will be created by querying all of the records from "todays_swan_file" that had "CENTERA" assigned as the attributed_center.

```
PROC SQL;
  CREATE TABLE work.SWAN_&center._01 AS
  SELECT *
  FROM todays_swan_file
  WHERE upper(attributed_center) = "&center";
QUIT;
```

Next, a conditional statement is executed to differentiate the processing required to find the primary care provider and last clinic appointment identified in the EMRs of Centers A & B (which use the same EMR vendor) from that required for Centers C, D, & E (which use a different EMR).

```
%if &center = CENTERA or
    &center = CENTERB or
```

```
%then %do;
```

First the subset of ADT records specific to the current center is matched to the EMR by the patient's Medicaid ID to get the patient's first and last name as it is listed in the emr.

```
PROC SQL;
CREATE TABLE swan_&center._02 AS
SELECT a.*,
       compress(b.last)||', '||compress(b.first) AS emr_patient_name,
       b.providerid
FROM work.swan_&center._01 a LEFT JOIN
     &center..patients b
ON input(a.medicaid_id,best8.) = b.medicaid_id;
QUIT;
```

Next, the patient's primary care provider's name is added using a query against the "provider's" table in the EMR after getting the patient's primary care provider id from the previous query. Note, these two steps could certainly be collapsed into one (more complex) query, but they are kept as two separate queries here because the number of records (and thus the size of the data files) is very small and it makes the code a bit simpler to follow.

```
PROC SQL;
CREATE TABLE swan_&center._03 AS
SELECT a.*,compress(b.last)||', '||compress(b.first) AS PCP
FROM swan_&center._02 a LEFT JOIN
     &center..providers b
ON a.providerid = b.providerid;
QUIT;
```

Similarly, a dataset of the most recent patient encounters for the current set of patients is created using PROC SQL and this list of most-recent encounters is added to the patient discharge summary records. Note that as the "swan_¢er_xx" datasets are created, they are created with LEFT Joins from the base dataset—as some of the attributed members will not have an entry in the EMR database (for example, if they have never been seen as a patient at their assigned primary care physician).

```
PROC SQL;
CREATE TABLE work.swan_&center._encounters AS
SELECT patientprofileid, max(put(datepart(visit), yymmdd10.)) AS
       last_encounter_date
FROM &center..Visits
WHERE patientid in (SELECT patientid
                   FROM swan_&center._01)
GROUP BY patientprofileid;
QUIT;
```

```
PROC SQL;
CREATE TABLE work.swan_&center._04 AS
SELECT a.*,b.last_encounter_date
FROM swan_&center._03 a LEFT JOIN
     work.swan_&center._encounters b
ON input(a.patientid) = b.patientprofileid;
QUIT;
```

```
%END;
%ELSE %DO;
```

Following the addition of details from the EMR to the ADT records for Centers A & B, similar logic is provided in the macro to cover the cases in which the macro is called for processing of data for Centers C, D, & E. Again, these

centers utilize a different EMR, with a different data model, so when the macro is executed for these centers, the previous processing steps will be skipped and processing of those centers's data will begin here.

[Equivalent Processing for Centers C, D, & E against their EMR Data Model]
 %END;

After the conditional processing (for Centers A/B vs. C/D/E) is finished, the resulting "swan_¢er_04" dataset is cleaned up (creating null values for the PCP variable and dropping the EMR's internal "doctorid" (which the clinic staff will not actually recognize as a useful data element).

```
DATA work.swan_&center._final;
  SET work.swan_&center._04;
  IF compress(pcp) = ',' THEN pcp = '';
  DROP doctorid;
RUN;
```

This final dataset is then prepared for delivery to the appropriate center as an Excel file. The file is dynamically named by creating the macro variable "swan_distrib_file" with the following DATA _NULL_ step. The value assigned to the filepath is "C:\SWAN\Output\" concatenated with the name of the center (as passed in the "center" parameter to the "swan_mail" macro, followed by a datetime stamp in the form of YYYYMMDDHHmmss.

```
DATA _NULL_
  CALL SYMPUT ('swan_distrib_file',
    "c:\SWAN\Output\&center._" ||
    compress(translate(put(date(), yymmdd10.), "", "-") ||
    translate(put(time(), tod.), "", ":")) || '.xls');
RUN;

PROC EXPORT DATA= work.swan_&center._final
  OUTFILE= "&swan_distrib_file"
  DBMS=excelcs REPLACE;
RUN;
```

Therefore, when the "swan_mail" macro is run with the center specified as "CENTERA", the PROC EXPORT step, above, will generate the report of ADT records for Center A to the following file:

c:\SWAN\Output\CENTERA_20170910193914.xls

With the file for Center A written out to Excel, the program begins generating VBScript syntax (Johnson, 2016) to send the encrypted e-mail with the current day's report attached. The first step is to create a macro variable that contains the e-mail addresses of the report recipients. Having read these data in from Excel previously in the program, a PROC SQL step reads the recipients into a list macro variable delimited with a semi-colon. Because the macro variable "CENTERA_recipients" will resolve to a text string containing semi-colons (e.g., jsmith@centera.org; mrollins@centera.org) and throw an error in the later DATA step, a second macro variable "sendto" is created that uses multiple ampersands so that the final resolution of the variable is the quoted value contained in the macro variable "CENTERA_recipients" (see Carpenter, 2004, for further discussion of the use of multiple ampersands).

```
PROC SQL NOPRINT;
  SELECT email INTO :&center._recipients separated BY '; '
  FROM swan_distrib_list
  WHERE center = symget('center') AND
    active='Y';
QUIT;

%LET sendto="&&&center._recipients";
```

Next, the .vbs file that will be used to execute and send the email is built. This portion of the program begins with assigning the location of the script. Note that each execution of the "swan_mail" macro will result in a unique VBScript identifying the center for which the VBScript is written as well as the execution date and time. This DATA

NULL step also creates the “rpt_date_mmddyy” macro variable in the form of MM/DD/YYYY to use in the subject line of the e-mail.

```
DATA _NULL_;
  CALL SYMPUT ('script_name',
    """"||"c:\SWAN\Output\scripts\SWAN_&center._mailer"||'-'||
    compress(translate(put(date(), yymmdd10.), "", "-"))||
    translate(put(time(), tod.), "", ":"))||'.vbs'||"""" );
  CALL symput ('rpt_date_mmddyy', put(date(), mmddyy10.));
RUN;
```

In addition to the list of recipients and the name of the VBScript file, three additional macro variables are generated to add additional information to the subject line of the e-mail—the “nickname” of the center from the “full_names” dataset created by the previous CARDS statement, a count of the number of ED discharges contained in the file, and a count of the number of IP discharges contained in the file.

```
PROC SQL NOPRINT;
  SELECT abbreviation INTO :center_abbreviation
    FROM work.full_names
    WHERE upper(nickname)=symget('center');
QUIT;

PROC SQL NOPRINT;
  SELECT count(*) INTO :ed_count
    FROM swan_&center._final
    WHERE patient_type = 'E';
QUIT;

PROC SQL NOPRINT;
  SELECT count(*) INTO :IP_count
    FROM swan_&center._final
    WHERE patient_type = 'I';
QUIT;
```

The VBScript file is written in a DATA _NULL_ step by issuing a series of “PUT” commands to write the VB syntax. The “to” line is completed with a reference to the “sendto” macro variable containing the semicolon-delimited list of e-mail recipients, the subject line is written using references to the date as well as to the macro variables containing the number of IP and ED discharges. It is important to note that the “SECURE” in the subject line is used by the organizations e-mail encryption software to recognize the email as one that needs to be encrypted before it is sent. Without this keyword, the e-mail would not be encrypted. Please check with your local security policies before utilizing this methodology for sending secure information such as personal health information.

```
FILENAME script &script_name;
DATA _NULL_;
FILE script;

PUT '''Set Variables';
PUT 'mailto = "&sendto"';
PUT 'subject = '''SECURE - SWAN Report
  (%SYSFUNC (COMPRESS (&rpt_date_mmddyy))): &center - ED:
  (%SYSFUNC (COMPRESS (&ed_count)) ) / IP: (%SYSFUNC (COMPRESS (&ip_count))
  )''';
```

Next, the attachment is specified by referencing the “swan_distrib_file” macro variable, and the body text is specified. The script syntax to create the email object and specify the values of the required parameters are assigned from the previously specified variables, and the send method is specified.

```
PUT 'attachment = '''&swan_distrib_file''';
```

```

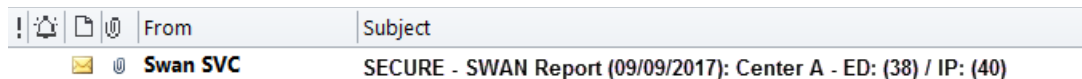
PUT 'body = "Attached is the Statewide Alert Notification (SWAN) Report
Delivered this morning from IHIN."';
PUT 'Email mailto, subject, body, attachment';
PUT 'Function to Generate Email';
PUT 'Sub Email (mailto, subject, body, attachment)';
PUT '    Set oapp = CreateObject("Outlook.Application")';
PUT '    Set oitem = oapp.CreateItem(0)';
PUT '    With oitem';
PUT '        .To = mailto';
PUT '        .Subject = subject';
PUT '        .HTMLBody = body';
PUT 'End With';
PUT 'Set msgattachments = oitem.Attachments';
PUT '    msgattachments.Add attachment';
PUT 'oitem.send';
PUT 'End Sub';
RUN;

X &script_name;

%MEND swan_mail;

```

With this script written, an X command is executed to call the script, and the e-mail is sent to the recipients at Center A.



Although the logic of the “swan_mail” macro was described using a macro call for Center A, the SWAN Distribution program is completed with multiple macro calls for each of the centers participating in the ACO. Each execution of the macro takes just a few seconds to complete so that by 7:30 a.m. the care coordination teams have the information they need to contact patients and schedule follow-up visits with their primary care provider, as appropriate.

```

%swan_mail(center=centera);
%swan_mail(center=centerb);
%swan_mail(center=centerc);
%swan_mail(center=centerd);
%swan_mail(center=centere);

```

```
PROC PRINTTO;
```

```
RUN;
```

CONCLUSIONS & NEXT STEPS

Without the automation of the processes described above, the ACO would need to manually download the ADT file, sort it, parse it into files for each clinic, and manually send emails to the care teams. Additionally, if one wanted to add the additional EMR information about each patient’s primary care provider, their last visit date, and their primary/secondary insurance coverages, this process could easily take a data coordinator the better part of a day to complete, by scheduling the program to run on Windows® Task Scheduler (Schacherer, 2013); , the process is completely automated and all that is required to add members of the care team to the distribution list is to add them to the Excel file that drives generation of the recipients list for each center’s e-mail. This approach has been used to solve multiple reporting challenges that have arisen while entering the world of value-based care. In addition to these data management and automation challenges, SAS has also provided tremendous value in measuring clinical quality, calculating patient risk & complexity, and measuring financial performance.

REFERENCES

- Carpenter, A. (2004). Carpenter's Complete Guide to the SAS® Macro Language, Second Edition. Cary, NC: SAS Institute, Inc.
- Johnson, C. (2016). Integrating Microsoft® VBScript and SAS®. Proceedings of SAS Global Forum 2016. Cary, NC: SAS Institute, Inc.
- National Transitions of Care Coalition (2017). <http://ntocc.org>.
- SAS (2008). TS-800 Configuring SSH Client Software in UNIX and Windows Environments for Use with the SFTP Access Method in SAS® 9.2, SAS® 9.3, and SAS® 9.4. (<https://support.sas.com/techsup/technote/ts800.pdf>). Cary, NC: SAS Institute, Inc.
- Schacherer, C. (2013). SAS® Data Management Techniques: Cleaning and transforming data for delivery of analytic datasets. Proceedings of SAS Global Forum 2013. Cary, NC: SAS Institute, Inc.
- Schacherer, C. (2012). The FILENAME Statement: Interacting with the world outside of SAS®. Proceedings of SAS Global Forum 2012. Cary, NC: SAS Institute, Inc.
- Schacherer, C. (2011). Base SAS® Methods for Building Dimensional Data Models. Proceedings of SAS Global Forum 2011. Cary, NC: SAS Institute, Inc.
- Schacherer, C. & Detry, M. (2010). PROC SQL: From Select to Pass-Through SQL. Proceedings of South Central SAS Users Group. Cary, NC: SAS Institute, Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Chris Schacherer, Ph.D.
Clinical Data Management Systems, LLC
E-mail: chris.schacherer@cdms-llc.com
Web: www.cdms-llc.com

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