PharmaSUG 2016 - Paper AD17

The New STREAM Procedure as a Virtual Medical Writer

Joseph Hinson, inVentiv Health, Princeton, NJ, USA

ABSTRACT

One of the really cool features of SAS® 9.4 is the STREAM procedure. This new tool allows free-form text containing macro elements to be directly streamed to an external file, completely bypassing the SAS® Compiler. The SAS® Word Scanner allows the Macro Processor to resolve the macro elements after which text and graphical elements are streamed directly to the external file without engaging the SAS® Compiler. This means SAS® syntax violators like HTML and XML tags would not trigger errors. This exception permits greater freedom in composing text for processing, and thus very much suited for reports such as patient narratives, patient profiles, reviewer's guide, and investigator's brochures. Such documents typically contain data in a variety of formats: textual paragraphs interspersed with bulleted lists, tables, images, graphs, schematics, and listings, all of which are easily incorporated by the use of macro elements like %includes, macro variables, and even macro calls. The key feature in Proc STREAM is an input stream of free text embedded with macro elements, where the macros are executed and expanded while the remaining text in the input stream is preserved and not validated as SAS® syntax. Such flexibility of PROC Stream is demonstrated in this paper with the creation of Patient Narratives and Investigator Brochures.

INTRODUCTION

Proc STREAM has much to do about how SAS® processes code sitting in the input stack. Normally, the Word Scanner examines and tokenizes the stream of characters from the input stack. Depending on the type of tokens, the word scanner triggers one of the following: the DATA Step Compiler, the Macro Processor, the Command Processor, or the SCL Compiler. The Macro Processor in particular is triggered by ampersand (&) or the percent (%) symbols. Proc STREAM allows various macro specifications to be embedded in arbitrary text such that only the Macro Processor is invoked while the rest of the text gets streamed directly to the external file. Without this selective processing by Proc STREAM, ordinary text containing macro triggers would have macro processing, but the DATA step compiler would flag syntax errors with the remaining text. The non-macro text may not be valid SAS® syntax. Typical examples are HTML and XML elements. By allowing text to be written as-is (except for macro specifications), it becomes much easier to compose specialized text as found in clinical trial patient narratives and investigator's brochures. As this paper will demonstrate, Proc STREAM allows the kind of versatility and flexibility that permits the inclusion of tables, figures, lists, HTML tags, macros, macro-variables in a clinical trial document.

The paper will present Proc STREAM in six parts:

1. The Proc STREAM syntax
2. Application to a simple narrative
3. The "Macro Call Variable" concept
4. Proc STREAM application to a complete patient narrative in HTML

5. Adapting Proc STREAM for RTF documents
6. How to embed RTF tables in Investigator's Brochures

PART 1: THE PROC STREAM SYNTAX

The Proc STREAM statement specifies an external file with the "OUTFILE=" keyword, as well as options.

The arbitrary text is wrapped inside a "BEGIN" and a four semi-colon ending ";;;;".

There is no "RUN" or "QUIT".

```
PROC STREAM OUTFILE=fileref <options>;
BEGIN
(some text which may contain macro triggers)
;;;;
```

Two Useful Proc STREAM statement Options:

a. **RESETDELIM=** "label"

The SAS® Word Scanner expects macro statements like "%LET" and "%INCLUDE" to begin on a statement boundary -- which means, the statements must be preceded by a semicolon and must end with a semicolon. Therefore to use %LET and %INCLUDE in a Proc STREAM input text, one must place a special marker token before the statements and end the statement with a semicolon. This special marker token is defined with the option RESETDELIM=*label*, where *label* can be any arbitrary SAS® name:

```
PROC STREAM  OUTFILE= myfile  RESETDELIM="goto";

BEGIN

goto; %INCLUDE myotherfile;

;;;;
```

The marker token specified by RESETDELIM is also required when a carriage return is required in the input text. The keyword "NEWLINE" is used with the marker token:

```
PROC STREAM  OUTFILE= myfile  RESETDELIM="goto";

BEGIN

Dear Sir, goto NEWLINE;
The profile below is for patient 12345.

;;;;
```

b. **QUOTING=SINGLE** *(or* **DOUBLE** *or* **BOTH***)*

This option specifies that the single quotation mark (') should be treated like any other character, as expected for: *the patient's blood pressure*.

QUOTING=DOUBLE would be required in cases where the text involves, for instance, XML elements in Define-XML

documents:

```
<ItemRef ItemOID="ADSL.ARM" OrderNumber="6" Mandatory="No"/>
```

PART 2: THE CONCEPT AND USE OF "MACRO CALL VARIABLES"

The possibility of embedding macro elements in a streaming text provides great opportunity for creating "smart text": text that can run tiny SAS® codes to select and insert certain values. Usually, for direct text substitutions, macro variables are more than adequate. However there are situations where some local processing is much more desirable. A "macro call variable" can be considered a small macro call that generates a macro variable on the spot, to be resolved and inserted in the streaming text.

(a) An example is choosing the pronouns "he" or "she" in a sentence. A "macro call variable" would execute a small code on the spot to determine the gender and select "he" or "she" for insertion. Such calls have a characteristic missing semi-colon in the macro statements (see the yellow-highlighted part below) containing the derived macro variable ***&heshe***:

```
%let vv=Male;

%macro hsh();
%global heshe;
%if "&vv." eq "Female" %then %let heshe=She;
%else %if "&vv." eq "Male" %then %let heshe=He;
&heshe
%mend hsh;
```

As applied in streaming text:

```
The patient was a 149-year old Martian male.  %hsh() was diagnosed with atrial fibrillation.
```

(b) A more elaborate example is obtaining a variable's value from a look-up table, for example, *patprofile*:

The patprofile dataset would contain parameter name/value pairs for each *usubjid*. The table is put in a hash object for look-up. When given the *usubjid* and the parameter name, the parameter value is returned. The entire code becomes a character argument for the DOSUBL function, which is then called by a macro.

```
%let codetext=%nrstr(data _null_;
                     if(1=2) then set patprofile;
                     declare hash pf(dataset:'patprofile');
                     rc=pf.defineKey('usubjid','paramname');
                     rc=pf.defineData('paramval');
                     rc=pf.defineDone();
                     rcc=pf.check(key:"&patient.",key:"&pname.");
                     if rcc eq 0 then do;
                     rcf=pf.find(key:"&patient.",key:"&pname.");
                     end;call symputx("pv",paramval,"g");run;);

%macro gp(pname);
%let x=%sysfunc(dosubl(&codetext));
&pv
%mend gp;
```

Note that the statement **&pv** has no semicolon, a hallmark of macro call variables. This feature also permits an error-free insertion into Proc STREAM.

The macro call variable *%gp* can then be used to look up and insert demographic information:

```
This %gp(age)-year-old %gp(race) %gp(sex) was admitted to the %gp(studyid) clinical trial at the
%gp(siteid) study site on %gp(randdt).
```

So, instead of creating in advance macro variables for AGE, GENDER, RACE, etc, a single macro call variable (***%gp***) does a hash table look-up within streaming text to create a macro variable and insert a value on the spot. The present paper employs macro call variables to make the text processed by Proc STREAM as versatile as possible.

PART 3: USING PROC STREAM TO CREATE A SIMPLE PATIENT NARRATIVE

A basic example of Proc STREAM would consist of just simple text containing macro variables:

```
filename aefolder "C:\Users\admin\Desktop\SASoutputs\aenarrative.rtf" ;
PROC STREAM OUTFILE=aefolder;
BEGIN
Subject &patid. was a &age.-year old &race. &sex. from &country..  The subject consented to partake
in the &studyid. on &randdt. and was first exposed to study medication &trt.. The subject
 was &height. ft in height, &weight. lb in weight and suffered from &disease..
;;;;
```

OUTPUT-1:

```
Subject 001 was a 45-year old white female from Germany.  The subject
consented to partake in study A123 on 27Apr99 and was first exposed to
study medication CoolDrug. The subject was 6 ft in height, 165 lb in
weight and suffered from diabetes.
```

PART 4: CREATING A FULL-BLOWN PATIENT NARRATIVE IN HTML

A real narrative for a clinical trial subject might involve more than as depicted above. First, the layout might consist of brief lines of demographic data at the top, then free text with various substitutions, followed by a list of items for concomitant medication, medical history, adverse events, then a small table of non-serious adverse events, and lastly a figure image depicting the adverse events profile. To allow the inclusion of such varied output forms, the HTML destination is the most suitable choice. With HTML, even images are easily incorporated with the <img source……> element. In addition to using various HTML elements, the use of the previously-described "macro call variables" also provide an added versatility. So, not only would the streaming text contain macro variables but also actual macro calls, %Include statements, and html statements. Such versatility allows for the automatic determination of "He" versus "She", "her" versus "his", and macro variables that generate a list of items, one item per line, regardless of how varying the number of items are. A macro call can also invoke a large code like a table-creating program as well as a tiny HTML carriage return code. Furthermore, the streaming text can be embedded with graphical images through the use of HTML image statements.

Generating the HTML patient narrative is done below in a number of steps:

SECTION-A: Prepare Input Data

1. Create the main patient data set, patdata, containing demographic, adverse event, diagnoses, and concomitant medication information. Create the main patient data set, patdata, containing demographic, adverse event, diagnoses, and concomitant medication information.
2. Create the main patient data set, patdata, containing demographic, adverse event, diagnoses, and concomitant medication information.
3. Derive a table with parameter-value pairs from patdata. This table, patprofile, will be loaded in to a hash object as a look-up table for a value-getting macro called %gp.
4. Use ODS HTML and the REPORT Procedure to create a Concomitant Medication table to be inserted into Proc STREAM with a %include.

SECTION-B: Create Macro Elements

5. Create macro variables for HTML codes heading level is for a subtopic of a main topic. This is the paper body.
6. Convert lists of Adverse Events and Concomitant Medications into macro variables This is the paper body.
7. Define “macro call variable” %gp for looking up patient data the paper body. This is the paper body. This is 7
8. Define “macro call variable” %hsh for inserting ‘he’ or ‘she’ based on gender the paper body. This is the paper body.
9. Define “macro call variable” %hsr for inserting ‘his’ or ‘her’ based on gender.

SECTION-C: Run Proc STREAM

THE COMPLETE CODE:

```
*Step:1-------------------------------------------------------------------------
CREATE MAIN PATIENT DATA SET patdata
-------------------------------------------------------------------------------;

data patdata;
infile datalines;
input usubjid $ studyid $ siteid $ trt $ age $ race $ sex $ aept : $15. cat $ randdt : $10.
disease : $30. DiagDate : $10. ConMed : $30.;
datalines;
999004 ABC123 XY005 NewDrug 63 Asian Female Diarrhea Mild 2010-02-24 ColonCancer 2010-01-03
Ibuprofen
999005 ABC123 XY005 NewDrug 58 White Male Dyspepsia Mild 2010-03-12 LungCancer 2010-01-04
Clemastine
999006 ABC123 XY005 NewDrug 66 Black Male Epilepsy Serious 2010-03-28 BrainCancer 2010-01-05
Acetamenophen
999007 ABC123 XY005 NewDrug 61 Asian Female Arrhythmia Serious 2010-03-24 BoneCancer 2010-01-06
AlphaLipoicAcid
999007 ABC123 XY005 NewDrug 61 Asian Female Diarrhea Mild 2010-03-24 BrainCancer 2010-01-06
AlphaLipoicAcid
999007 ABC123 XY005 NewDrug 61 Asian Female Syncope Serious 2010-03-24 KidneyCancer 2010-01-06
AlphaLipoicAcid
999010 ABC123 XY005 NewDrug 65 White Male Diarrhea Mild 2010-05-24 BoneCancer 2010-01-09
NicotinamideRibose
999011 ABC123 XY005 NewDrug 74 White Female Constipation Mild 2010-06-24 ColonCancer 2010-11-07
Ibuprofen
999012 ABC123 XY005 NewDrug 53 White Female Migraine Serious 2010-02-24 BrainCancer 2010-02-08
Dexamethasone
999013 ABC123 XY005 NewDrug 62 White Male Delirium Serious 2010-07-24 BrainCancer 2010-01-09
Chloroquine
999014 ABC123 XY005 NewDrug 87 White Male Diarrhea Mild 2010-01-24 ColonCancer 2010-01-02
AlphaLipoicAcid
999015 ABC123 XY005 NewDrug 93 Asian Female Nausea Mild 2010-04-24 LungCancer 2010-01-08
Acetamenophen
999016 ABC123 XY005 NewDrug 49 White Female Hallucination Serious 2010-09-24 BrainCancer 2010-01-
04  Hydrochlorothiazide
;
run;
```

```
*Step:2-------------------------------------------------------------------------
DERIVE patprofile TABLE FROM PATDATA
-------------------------------------------------------------------------------;
data patprofile(keep=usubjid paramname paramval);
dsid=open("patdata");
nrows=attrn(dsid,"NOBS");
ncols=attrn(dsid,"NVARS");
do r=1 to nrows;
       dc=fetchobs(dsid,r);
       do c=2 to ncols;
       usubjid=getvarc(dsid,1);
       paramname=varname(dsid,c);
```

```
        test=vartype(dsid,c);
        if test="C" then paramval=getvarc(dsid,c);
        else if test="N" then paramval=put(getvarn(dsid,c),best.);
        output;
        end;
end;
dc=close(dsid);
run;
```

```
*Step:3-----------------------------------------------------------------------
USE ODS HTML AND PROC REPORT TO GENERATE CONCOMITANT MEDICATIONS TABLE
-------------------------------------------------------------------------------;
%let outputlocation=C:\Users\admin\Desktop\SASoutputs\conmedtable.rtf;

        options nonumber nodate nocenter linesize=max pagesize=min;
               title;*<----to suppress system title;
        ods listing close;
        ods escapechar="^";
               ods rtf file="&outputlocation." ;
               proc report data=patdata(where=(usubjid eq '999005')) split="*";
               column trt usubjid diagdate race sex age disease conmed;
               define trt/display 'Treatment*Arm';
               define usubjid/display 'Patient*ID';
               define diagdate/display 'Diagnosis*Date';
               define race/display 'Race';
               define sex/display 'Sex';
               define age/display 'Age';
               define disease/display 'Disease';
               define conmed/display 'Con Meds';
               run;
        ods html close;
        ods listing;
```

```
*Step:4-----------------------------------------------------------------------
CREATE MACRO VARIABLES FOR HTML TEXT STYLING AND FOR SELECTED PATIENT
-------------------------------------------------------------------------------;
%let patient=999005;
%let z=%str(<br/>);*<------for carriage return within HTML;
%let y=%nrstr(&#8226);*<-----for the bullet symbol within HTML;
%let nbsp=%nrstr( );*<----to prevent   resolving warning in LOG;

*Step:5-----------------------------------------------------------------------
CONVERT LISTS OF ADVERSE EVENTS AND CONMED INTO MACRO VARIABLES
-------------------------------------------------------------------------------;
proc sql noprint;
select distinct aept into :aetexts separated by "&z.&y."
from patdata
where cat eq "Serious";
quit;

proc sql noprint;
select distinct ConMed into :cmtexts separated by "&z"
from patdata;
quit;
```

```
*Step:6-----------------------------------------------------------------------
          DEFINE MACRO CALL VARIABLE gp FOR LOOKING UP PATIENT DATA
-------------------------------------------------------------------------------;
%let codetext=
  %nrstr(data _null_;
  if(1=2) then set patprofile;
  declare hash pf(dataset:'patprofile');
  rc=pf.defineKey('usubjid','paramname');
  rc=pf.defineData('paramval');
  rc=pf.defineDone();
  rcc=pf.check(key:"&patient.",key:"&pname.");
```

```
  if rcc eq 0 then do;
  rcf=pf.find(key:"&patient.",key:"&pname.");
  end;call symputx("pv",paramval,"g");run;);

%macro gp(pname);
%let x=%sysfunc(dosubl(&codetext));
&pv.
%mend gp;
```

```
*Step:7-----------------------------------------------------------------------------
    DEFINE MACRO CALL VARIABLE hsh FOR INSERTING he OR she BASED ON GENDER
-------------------------------------------------------------------------------------;
%let vv=%gp(sex);

%macro hsh();
%global heshe;
%if "&vv." eq "Female" %then %let heshe=She;
%else %if "&vv." eq "Male" %then %let heshe=He;
&heshe
%mend hsh;
```

```
*Step:8-----------------------------------------------------------------------------
    DEFINE MACRO CALL VARIABLE hsh FOR INSERTING his OR her BASED ON GENDER
-------------------------------------------------------------------------------------;
%macro hsr();
%global hisher;
%if "&vv." eq "Female" %then %let hisher=her;
%else %if "&vv." eq "Male" %then %let hisher=his;
&hisher
%mend hsr;
```

```
*Step:9-----------------------------------------------------------------------------
                              RUN PROC STREAM
-------------------------------------------------------------------------------------;
ods html close;
filename product "C:\Users\admin\Desktop\SASoutputs\patnarrative.html";
filename tabout "C:\Users\admin\Desktop\SASoutputs\conmedtable.html";
filename figout "C:\Users\admin\Desktop\SASoutputs\aefigure.html";

proc stream outfile=product resetdelim='goto';
BEGIN
Patient No.: &patient. &z.
Treatment Arm: %gp(trt) &z.
Demographics: %gp(age)-year-old %gp(race) %gp(sex) &z.
&z.&z.
This %gp(age)-year-old %gp(race) %gp(sex) was admitted to the %gp(studyid) clinical trial at the
%gp(siteid) study site on %gp(randdt).
&z.
%hsh() was diagnosed with %gp(disease) on %gp(DiagDate) by a team of oncologists at site
%gp(siteid)&z.
%hsh() exhibited, during the course of the trial, the following serious disorders:
&z.&y.&aetexts.. &z.
%hsh() had been taking before and during the study, some non-study medication as shown in the
table below: &z.
goto; %include tabout; &z.
The figure below summarizes all %hsr() non-serious adverse events (Placebo vs Multivitamins vs
NewDrug): &z.
goto; <img src="C:\Users\admin\Desktop\SASoutputs\adverseevents.jpg" alt="AE Figure"
style="width:690 px; height: 204 px">;
;;;;
```

Patient No.: 999005
Treatment Arm: NewDrug
Demographics: 58-year-old White Male

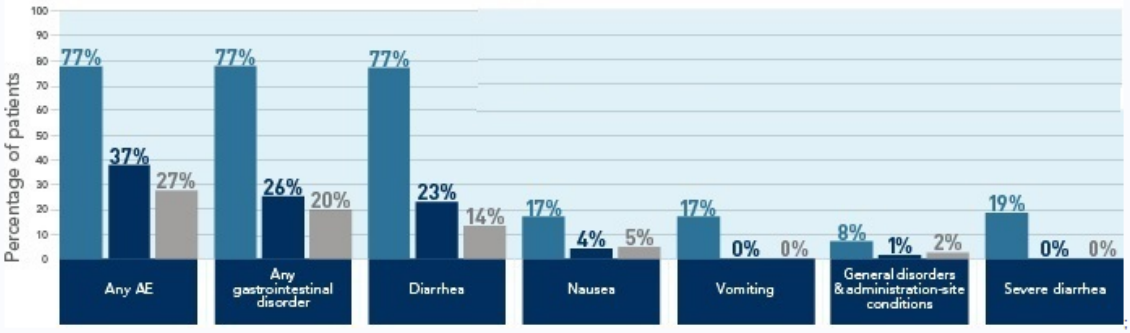
This 58-year-old White Male was admitted to the ABC123 clinical trial at the XY005 study site on 2010-03-12.
He was diagnosed with LungCancer on 2010-01-04 by a team of oncologists at site XY005
He exhibited, during the course of the trial, the following serious disorders:
•Arrhythmia
•Delirium
•Epilepsy
•Hallucination
•Migraine
•Syncope.
He had been taking before and during the study, some non-study medication as shown in the table below:

CONCOMITANT MEDICATIONS:

Treatment Arm	Patient ID	Diagnosis Date	Race	Sex	Age	Disease	Con Meds
NewDrug	999005	2010-01-04	White	Male	58	LungCancer	Clemastine

FOOTNOTE:...Generated with SAS version 9.4

The figure below summarizes all his non-serious adverse events (Placebo vs Multivitamins vs NewDrug):



Output 1. Output from a Full-Blown Proc STREAM Code for Patient Narratives

PART 5: ADAPTING PROC STREAM FOR RTF OUTPUTS

Generating RTF documents is a bit more challenging but feasible. The only drawback is that images cannot easily be inserted. However, text with table outputs can easily be generated. Formatting codes are issued with both Proc STREAM and RTF in mind. Such codes are also necessary for macro variable lists creation with Proc SQL. For carriage return, the RTF code "\par" is used.

```
%let z=%str(\par);*<------for carriage return within RTF;
%let y=%sysfunc(byte(219);*<-----for the bullet symbol within RTF;
%let nbsp=%sysfunc(byte(255);*<----to non-breaking space for RTF;
```

```
***(There MUST be space between " \par  and "%sysfunc(byte......");
proc sql noprint;
    select distinct aept into :aetexts separated by " \par %sysfunc(byte(149))"
       from patdata
            where cat eq "Serious";
quit;

proc sql noprint;
    select distinct ConMed into :cmtexts separated by "&z"
       from patdata;
quit;
```

The header section of a patient narrative also requires formatting compatible with both RTF and Proc STREAM. The code segment "%sysfunc(byte(149)) goto;" is required for proper positioning and alignment of header text. The bulleted list of adverse events provided by the macro variable &aetexts must be separated by the RTF code "\par".

```
filename product "C:\Users\admin\Desktop\SASoutputs\patientnarrative.rtf";
filename tabout "C:\Users\admin\Desktop\SASoutputs\conmedtable.rtf";

proc stream outfile=product resetdelim='goto';
BEGIN
%sysfunc(byte(149)) goto; Patient No.: &patient.
%sysfunc(byte(149)) goto; Treatment Arm: %gp(trt)
%sysfunc(byte(149)) goto; Demographics: %gp(age)-year-old %gp(race) %gp(sex)

This %gp(age)-year-old %gp(race) %gp(sex) was admitted to the %gp(studyid) clinical trial at the
%gp(siteid) study site on %gp(randdt).
&z.
%hsh() was diagnosed with %gp(disease) on %gp(DiagDate) by a team of oncologists at site
%gp(siteid)
%hsh() exhibited, during the course of the trial, the following serious disorders:
%sysfunc(byte(149)) goto;&aetexts
%hsh() had been taking before and during the study, some non-study medication as shown in the
table below:
;;;;
```

The use of Proc STREAM to generate RTF outputs with embedded tables require an approach quite different from HTML processing. Don Henderson[3] has single-handedly made this possible by developing a macro, %generateRTFTable, which provides the proper RTF header and table assembling codes. The next section describes the steps necessary for accomplishing that.

PART 6: HOW TO EMBED RTF TABLES IN INVESTIGATOR'S BROCHURES USING PROC STREAM

The whole idea is to put a macro call, *%generateRTFTable*, within an RTF template and put that template as an %include inside Proc STREAM:

a. Create an RTF template, e.g. template for Investigator's Brochure.
b. Insert *%generateRTFTable* macro calls at the desired locations within the template. Provide the desired SAS® data sets as parameters for the macros, as shown in Figure 1 below.
c. Run the macro definition for %generateRTFTable *(see Appendix).*
d. Run Proc STREAM, with IB_template.rtf as an %include:

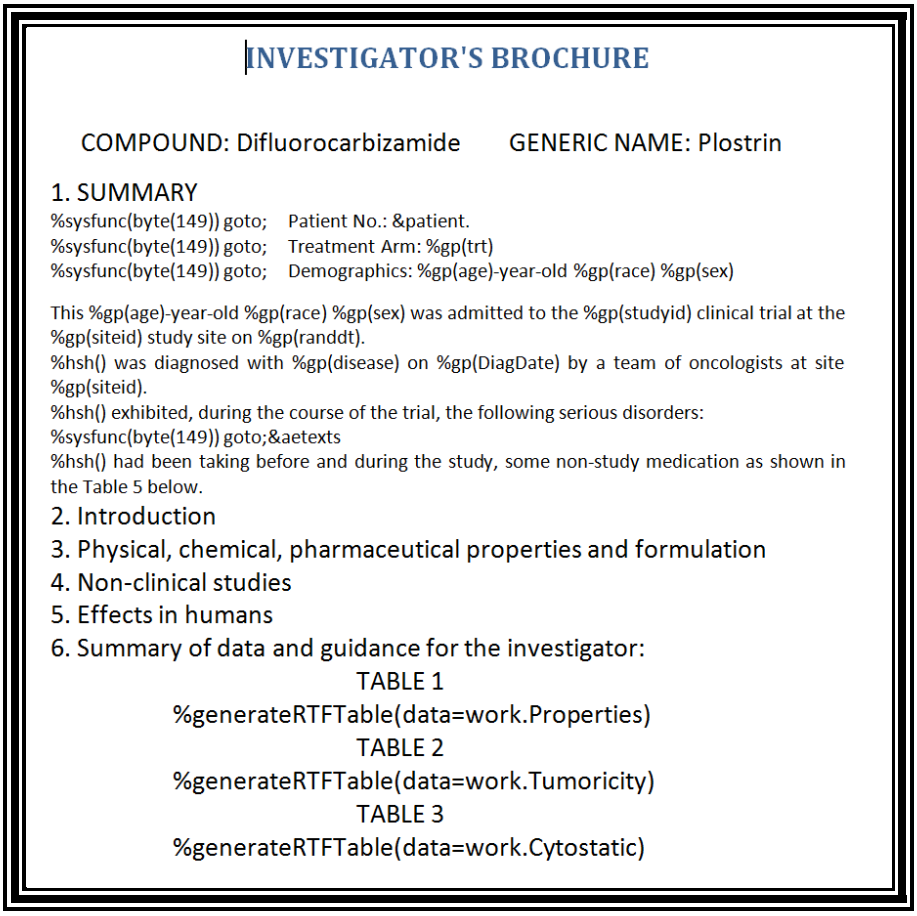


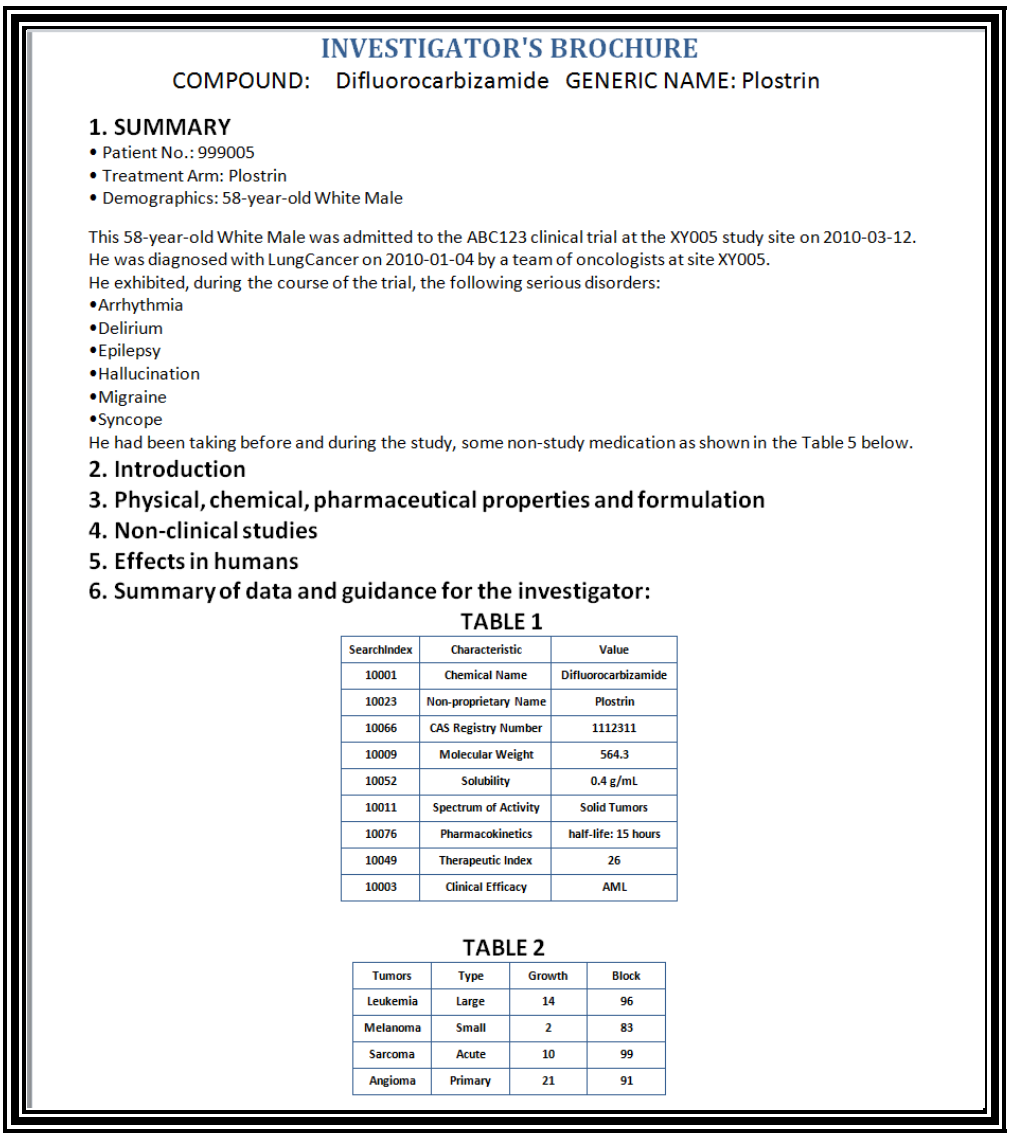
Figure 1. Template ("IB_template.rtf") for the creation of Investigator's Brochure

```
%let path = %nrstr(C:\Users\admin\Desktop\ProcStreamRTF\Templates); /* update with your own */

filename ibinfo "&path.\ib_template.rtf" lrecl = 32755;
filename out "C:\Users\admin\Desktop\SASoutputs\InvestBrochure.rtf" lrecl = 32755;


proc stream outfile = out quoting = single  resetdelim="goto";
BEGIN goto; %include ibinfo;
;;;;
```

When the above Proc STREAM code is run, the macro calls inside the investigator's brochure template would be resolved as tables, as shown in Output 3 below:



INVESTIGATOR'S BROCHURE

COMPOUND: Difluorocarbizamide GENERIC NAME: Plostrin

1. SUMMARY

- Patient No.: 999005
- Treatment Arm: Plostrin
- Demographics: 58-year-old White Male

This 58-year-old White Male was admitted to the ABC123 clinical trial at the XY005 study site on 2010-03-12.
He was diagnosed with LungCancer on 2010-01-04 by a team of oncologists at site XY005.
He exhibited, during the course of the trial, the following serious disorders:

- Arrhythmia
- Delirium
- Epilepsy
- Hallucination
- Migraine
- Syncope

He had been taking before and during the study, some non-study medication as shown in the Table 5 below.

2. Introduction

3. Physical, chemical, pharmaceutical properties and formulation

4. Non-clinical studies

5. Effects in humans

6. Summary of data and guidance for the investigator:

TABLE 1

SearchIndex	Characteristic	Value
10001	Chemical Name	Difluorocarbizamide
10023	Non-proprietary Name	Plostrin
10066	CAS Registry Number	1112311
10009	Molecular Weight	564.3
10052	Solubility	0.4 g/mL
10011	Spectrum of Activity	Solid Tumors
10076	Pharmacokinetics	half-life: 15 hours
10049	Therapeutic Index	26
10003	Clinical Efficacy	AML

TABLE 2

Tumors	Type	Growth	Block
Leukemia	Large	14	96
Melanoma	Small	2	83
Sarcoma	Acute	10	99
Angioma	Primary	21	91

Output 3. A Segment of Investigator's Brochure Showing Embedded Tables

CONCLUSION

The STREAM procedure allows documents to be mixed with SAS® macro elements without fear of triggering syntax errors. Thus a diverse type of information can be embedded without resorting to cutting and pasting. An RTF template with standard text can be created in advance, and macro elements added for automatic customization, as shown Figure 1. HTML outputs seem to be the most flexible of all, permitting any kind of embedded element. PDF, being binary, is not yet possible with Proc STREAM. However, conversions can easily be done from HTML or RTF.

RECOMMENDED READING

1. SAS® Institute, "STREAM Procedure" http://support.sas.com/documentation/cdl/en/proc/67327/HTML/default/viewer.htm#n12zrkr08eiacmn17lcv4fmt79tb.htm
2. Henderson, Don "PROC STREAM and SAS® Server Pages: Generating Custom HTML Reports" http://support.sas.com/resources/papers/proceedings14/1738-2014.pdf

3. Henderson, Don “A SAS® Server Page Approach to inserting SAS® Data Tables into RTF Documents" http://www.sascommunity.org/planet/blog/category/rtf-output/
4. Langston, Rick “Submitting SAS® Code On The Side” https://support.sas.com/resources/papers/proceedings13/032-2013.pdf

ACKNOWLEDGMENTS

The author is very grateful to his mentor, teacher, and manager, Brian Shilling. The author would also like to acknowledge Don Henderson for making available a very useful RTF macro, as well as the various educational publications about Proc STREAM.

CONTACT INFORMATION

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

Joseph W. Hinson, PhD
inVentiv Health
202 Carnegie Center, Suite 200
Princeton, NJ, 08540
1-609-282-1615
joehinson@outlook.com

APPENDIX (Don Henderson’s RTF Macro)

```
%macro generateRTFTable
      (data=sashelp.class
      ,where=
      ,vars=
      ,width=1000
          );

 /*-------------------------------------------------------------------------------------------
   Copyright (c) 2013 Henderson Consulting Services
   PROGRAMMER   : Don Henderson
   PURPOSE      : Reads the indicated data set and creates an RTF table.

 |-------------------------------------------------------------------------------------------|
 |  MAINTENANCE HISTORY                                                                      |
 |-------------------------------------------------------------------------------------------|
 |  DATE    |     BY    | DESCRIPTION                                                        |
 |----------|-----------|--------------------------------------------------------------------|
 | Feb 2013 |  Don H    | Original Development
 |-------------------------------------------------------------------------------------------*/

 %local dsid i ncolumns var rc numObs vnums vtypes vlabels;

 %if %length(&where) ne 0 %then %let where = (where = (&where));

 %let dsid = %sysfunc(open(&data&where));
 %if &dsid gt 0 %then
 %do;  /* have data */
    %let numObs = %sysfunc(attrn(&dsid,nobs));
    %if %length(&vars) = 0 %then
    %do;  /* display all the columns */
        %let nColumns = %sysfunc(attrn(&dsid,nvars));
        %do i = 1 %to &nColumns;
```

```
           %local vnum&i vtype&i vname&i vlabel&i vfmt&i vjust&i;
           %let vnum&i = &i;
           %let vtype&i = %sysfunc(vartype(&dsid,&&vnum&i));
           %if &&vtype&i = c %then %let vjust&i = l;
           %else %let vjust&i = r;
           %let var = %sysfunc(varname(&dsid,&&vnum&i));
           %let vlabel&i = %sysfunc(varlabel(&dsid,&&vnum&i));
           %let vlabel&i = %sysfunc(coalesceC(&&vlabel&i,&var));
           %let vfmt&i = %sysfunc(varformat(&dsid,&&vnum&i));
        %end;
    %end; /* display all the columns */
    %else
    %do;  /* display the specified columns */
        %let i = 1;
        %let var = %scan(&vars,&i,%str( ));
        %do %while(%length(&var) gt 0);
           %local vnum&i vtype&i width&i vlabel&i vfmt&i vjust&i;
           %let vnum&i = %sysfunc(varnum(&dsid,&var));
           %let vtype&i = %sysfunc(vartype(&dsid,&&vnum&i));
           %if &&vtype&i = c %then %let vjust&i = l;
           %else %let vjust&i = r;
           %let vlabel&i = %sysfunc(varlabel(&dsid,&&vnum&i));
           %let vlabel&i = %sysfunc(coalesceC(&&vlabel&i,&var));
                 %let vfmt&i = %sysfunc(varformat(&dsid,&&vnum&i));
           %let i = %eval(&i+1);
           %let var = %scan(&vars,&i,%str( ));
        %end;
        %let nColumns = %eval(&i-1);
     %end; /* display the specified columns */

         /* display the header row */
goto newline;{\par \trowd\trkeep\trhdr\trqc
     %do i = 1 %to &nColumns-1;
goto
newline;\clbrdrt\brdrs\brdrw10\brdrcf1\clbrdrb\brdrs\brdrw10\brdrcf1\clbrdrl\brdrs\brdrw10\brdrcf
1\cltxlrtb\clvertalb\clcbpat8\clpadt67\clpadft3\clpadr67\clpadfr3\cellx%eval(&i*&width)
     %end;
goto
newline;\clbrdrt\brdrs\brdrw10\brdrcf1\clbrdrb\brdrs\brdrw10\brdrcf1\clbrdrl\brdrs\brdrw10\brdrcf
1\clbrdrr\brdrs\brdrw10\brdrcf1\cltxlrtb\clvertalb\clcbpat8\clpadt67\clpadft3\clpadr67\clpadfr3\c
ellx%eval(&ncolumns*&width)
     %do i = 1 %to &ncolumns;
goto newline;\pard\plain\intbl\keepn\sb67\sa67\qc\f1\fs16\cf1{%qtrim(&&vlabel&i)\cell}
     %end;

     /* display the data rows */
goto newline;{\row}
     %do %while(%sysfunc(fetch(&dsid)) = 0);
goto newline;\trowd\trkeep\trqc
        %do i = 1 %to &nColumns-1;
goto
newline;\clbrdrb\brdrs\brdrw10\brdrcf1\clbrdrl\brdrs\brdrw10\brdrcf1\cltxlrtb\clvertalt\clcbpat8\
clpadt67\clpadft3\clpadr67\clpadfr3\cellx%eval(&i*&width)
        %end;
goto
newline;\clbrdrb\brdrs\brdrw10\brdrcf1\clbrdrl\brdrs\brdrw10\brdrcf1\clbrdrr\brdrs\brdrw10\brdrcf
1\cltxlrtb\clvertalt\clcbpat8\clpadt67\clpadft3\clpadr67\clpadfr3\cellx%eval(&ncolumns*&width)
        %do i = 1 %to &nColumns;
goto
newline;\pard\plain\intbl\sb67\sa67\qc\f1\fs16\cf1{%qsysfunc(getvar&&vtype&i(&dsid,&&vnum&i),&&vf
mt&i)\cell}
        %end;
goto newline;{\row}
    %end;
    %let dsid = %sysfunc(close(&dsid));
goto newline;\pard}
 %end; /* have data */

%mend generateRTFTable;
```