Paper 273-2012

ODS DOCUMENT From Scratch

Kevin D. Smith, SAS Institute Inc., Cary, NC

ABSTRACT

The ODS DOCUMENT destination is likely the most underutilized feature of ODS. The ODS DOCUMENT destination gives you the ability to customize the structure of your reports in ways that no other ODS features can. It enables you to store the actual ODS objects that are created when running a report, which you can then later rerun without invoking the procedures from the original report. You aren't limited to simply regenerating the same report; you can change the order in which objects are rendered, the table of contents, the templates that are used, macro variables, and system options. Even if you don't consider yourself an ODS wizard, don't be afraid to take a peek at the features of the ODS DOCUMENT destination in this paper because we'll be taking it from scratch.

INTRODUCTION

The ODS DOCUMENT destination may be the most underutilized tool in the ODS toolbox. This is probably due to the fact that it does not create visual output like PDF or HTML; it simply stores the information used by ODS to recreate a report. This makes the ODS DOCUMENT destination more abstract than most of the other ODS output types. While the ODS DOCUMENT destination does not create any output that you can view directly, it adds a whole new level of power to creating and modifying your reports. ODS PDF and ODS HTML destinations might be capable of generating nice looking output, but they do not give you much in the way of structuring and organizing the components of the report after the procedure has done its job. This is where ODS DOCUMENT comes in.

The ODS DOCUMENT destination adds a layer of customizability between the procedure (or DATA step) and ODS output types like PDF, HTML, RTF, etc. It doesn't create output that you view directly. It does allow you, with the help of the DOCUMENT procedure, to store and rearrange the individual components of a report, and change styles and options before the procedure output gets rendered. This gives you the ability to create customized versions of a report for different audiences without having to rerun the analytical procedures to get those report variants. In addition, the ODS DOCUMENT destination and the DOCUMENT procedure enable you to do things in ODS reports that are not possible through any other means, such as completely changing the report table of contents displayed by PDF, HTML, and RTF documents, modifying notes, and changing page breaks.

While there are advanced uses for the ODS DOCUMENT destination, getting started with the basics should feel natural to most ODS users. Since this paper promises to deal with ODS DOCUMENT "from scratch", we'll begin there and work up to more advanced uses as we progress through this paper.

For the remainder of the paper, "ODS DOCUMENT destination" is referred to as "ODS DOCUMENT". "ODS DOCUMENT" is also a SAS statement. "ODS document" refers to the actual file that is created by the ODS DOCUMENT destination.

All examples in the paper are repeated at the end of the paper without descriptive text between statements.

THE BASICS OF ODS DOCUMENT

Using ODS DOCUMENT is not any different than using other ODS output types such as PDF, RTF, or HTML. It uses the same "ODS sandwich" technique of opening the DOCUMENT destination with one ODS statement and closing the DOCUMENT destination with an ODS CLOSE statement. All of the procedure and DATA step code that is between those two statements will have their output stored in an ODS document. Here is a simple example. We have included the ODS HTML statement as well so that you can see what the rendered report looks like.

ODS Document From Scratch, continued

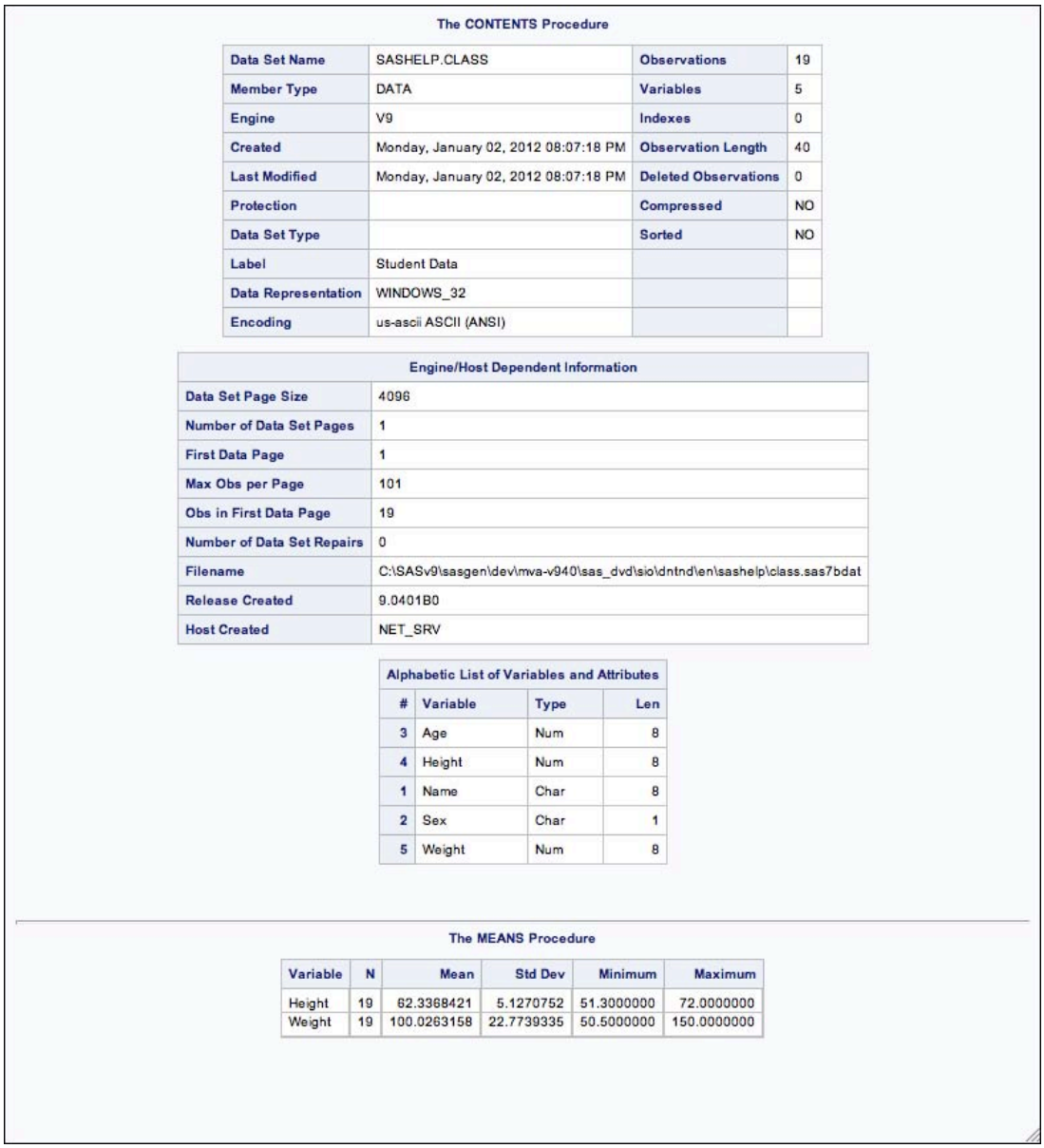
```
ods html file='mydocument.html';
ods document name=mydocument;

proc contents data=sashelp.class;
run;

proc means data=sashelp.class;
   var height weight;
run;

ods document close;
ods html close;
```

You will notice in the code above that the ODS DOCUMENT statement is very much like the other output type ODS statements, such as ODS PDF and ODS HTML. However, rather than giving a filename for the output, you give the name of an ODS document. This is because the ODS document file is in a binary format that is readable only by ODS DOCUMENT and PROC DOCUMENT, which must exist in a SAS library. By default, documents are saved to the Work library, but you can use a two level name to also specify a permanent library name (e.g., name=sasuser.mydocument).

By default, if you specify the name of a document that already exists, the new output will be appended to the existing document. This is called Update mode. When you are first starting out with ODS DOCUMENT, it's a good idea to use Write mode while you are iterating through your report-creating process. The default mode will always append to an existing document, which can cause confusion if you are writing to the same document multiple times in one session. You can overwrite a document by specifying Write mode after the name of the document:

```
ods document name=mydocument(write);
```

Display 1 shows the HTML output from the code above.



The CONTENTS Procedure

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Monday, January 02, 2012 08:07:18 PM	Observation Length	40
Last Modified	Monday, January 02, 2012 08:07:18 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Student Data		
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Engine/Host Dependent Information	
Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	101
Obs in First Data Page	19
Number of Data Set Repairs	0
Filename	C:\SASv9\sasgen\dev\mva-v940\sas_dvd\sio\dntnd\en\sashelp\class.sas7bdat
Release Created	9.0401B0
Host Created	NET_SRV

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Height	19	62.3368421	5.1270752	51.3000000	72.0000000
Weight	19	100.0263158	22.7739335	50.5000000	150.0000000

Display 1. HTML Output from ODS DOCUMENT Sample Code

We mentioned earlier that ODS DOCUMENT does not create output that is visual, it simply stores the output from procedures in a binary format; but how is that useful? The first and easiest use of ODS DOCUMENT is to replay a stored report. Before we do that, let's use PROC DOCUMENT to see what was stored in the report from the code above. The code below will list the full contents of a document:

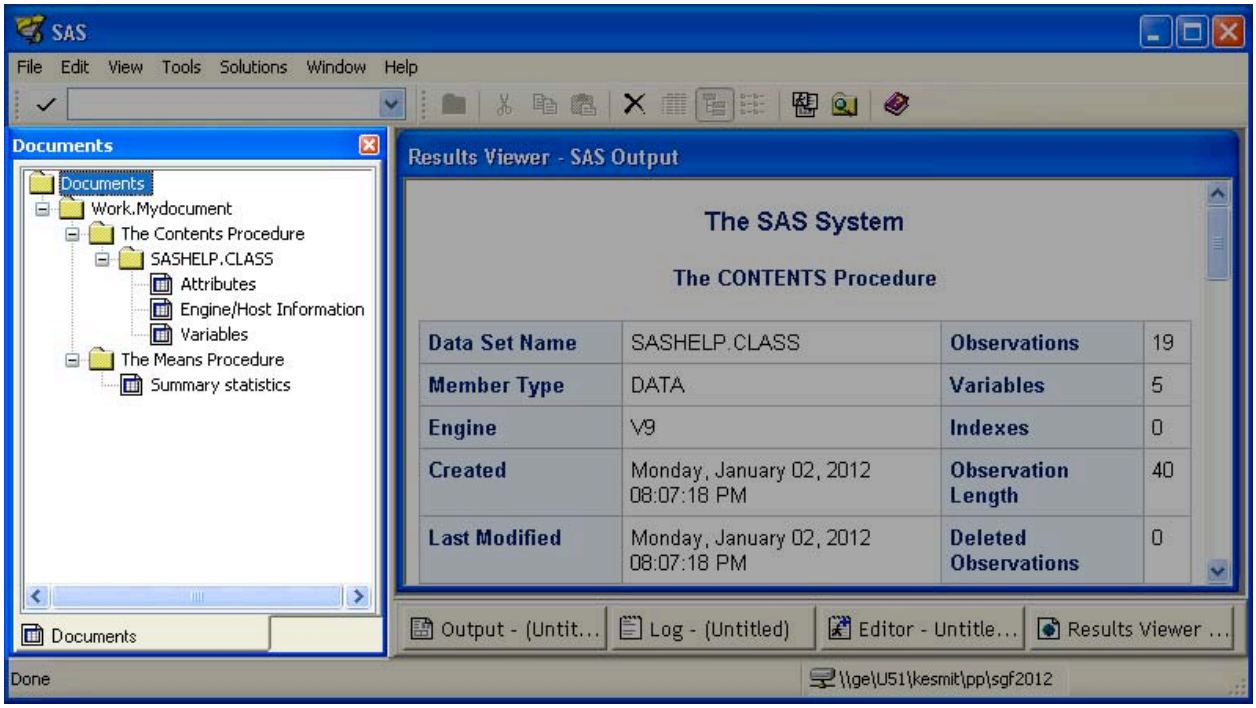
```
ods listing;
proc document name=mydocument;
   list / levels=all;
run;
ods listing close;
```

Here is the output from the LISTING destination[1]:

```
Listing of: \Work.Mydocument\
Order by: Insertion
Number of levels: All

  Obs    Path                                                                Type
-------------------------------------------------------------------------------
       1 \Contents#1                                                         Dir
       2 \Contents#1\DataSet#1                                               Dir
       3 \Contents#1\DataSet#1\Attributes#1                                  Table
       4 \Contents#1\DataSet#1\EngineHost#1                                  Table
       5 \Contents#1\DataSet#1\Variables#1                                   Table
       6 \Means#1                                                            Dir
       7 \Means#1\Summary#1                                                  Table
```

Output 1. Output from the PROC DOCUMENT LIST Statement

As you can see in Output 1, there are various types of items that are stored in the document. They include Dirs and Tables. Dirs are directories (or folders) and Tables are tabular output. The four pieces of tabular output, more generally called output objects, are evident from Display 1. There is another way to view the contents of a document as well. It is called the Documents window. Within an interactive SAS session, you can invoke the Documents window by entering ODSDOCUMENTS in the command bar. This will open a window that is like Windows Explorer, where you can navigate your documents by point-and-click. Display 2 shows the Documents window for our example.



Display 2. The Documents Window

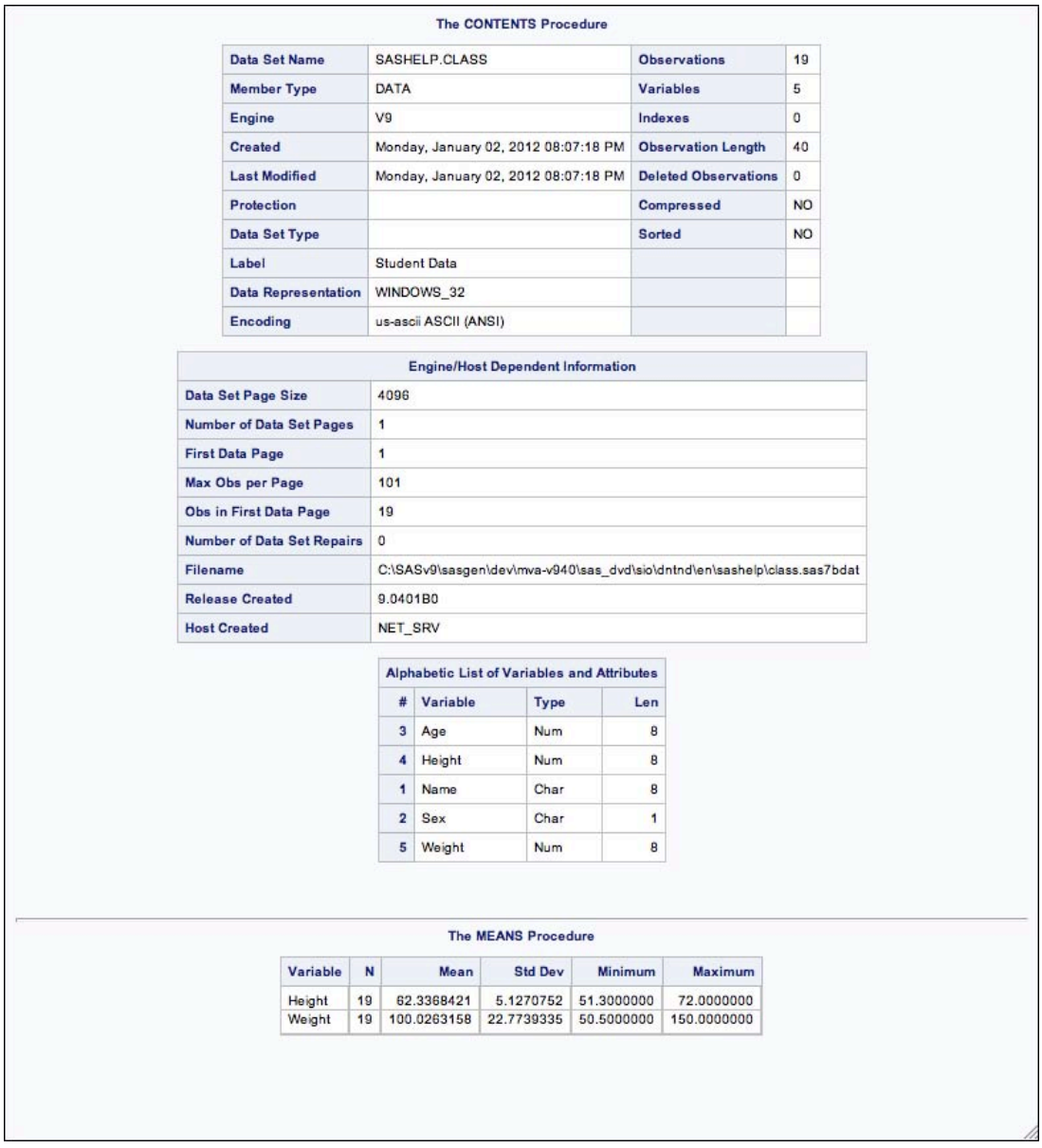
Now that we've verified that our document has been created, let's use PROC DOCUMENT to read the information from mydocument and replay our report. This is done with the REPLAY statement. When PROC DOCUMENT replays a report, it essentially does the same operations that procedures use to tell ODS what to render. However, rather than processing data sets and constructing output objects from the analyses, PROC DOCUMENT simply reads the information stored in the document and constructs output objects from that. The output objects are sent off to

[1] The table created by the LIST statement is an output object, so it is rendered to all open ODS destinations.

ODS, which renders them just as if the procedure was the one to send the information. The code below will replay our stored report to the HTML destination without doing a reanalysis.

```
ods html file='replay.html';
proc document name=mydocument;
   replay;
run;
ods html close;
```

Display 3 shows the HTML output from replaying the document:



The CONTENTS Procedure

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Monday, January 02, 2012 08:07:18 PM	Observation Length	40
Last Modified	Monday, January 02, 2012 08:07:18 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Student Data		
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Engine/Host Dependent Information	
Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	101
Obs in First Data Page	19
Number of Data Set Repairs	0
Filename	C:\SASv9\sasgen\dev\mva-v940\sas_dvd\sio\dntnd\en\sashelp\class.sas7bdat
Release Created	9.0401B0
Host Created	NET_SRV

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Height	19	62.3368421	5.1270752	51.3000000	72.0000000
Weight	19	100.0263158	22.7739335	50.5000000	150.0000000

Display 3. HTML Output from the PROC DOCUMENT REPLAY Statement

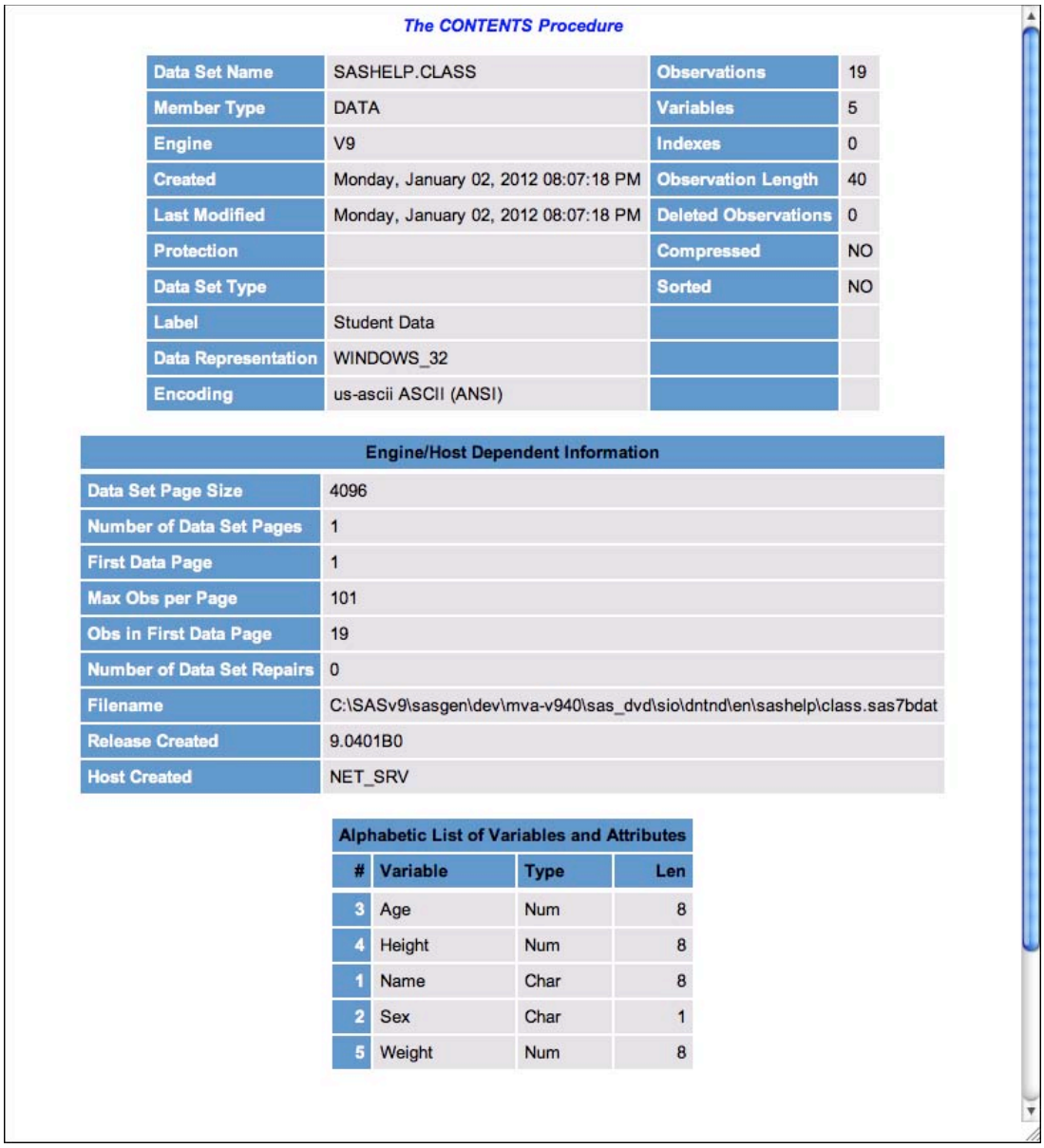
As you can see, the reports in Display 1 and Display 3 are identical. This is a trivial use of ODS DOCUMENT, but it does have some real world uses if the analysis used in the original report are time consuming or processor intensive. Let's say that you worked for some time to get all of the information in the report that you needed, but running the report takes a long time. You had been generating HTML versions of the report all along, but now your boss tells you that he/she wants a printed version. Without ODS DOCUMENT, you would have to add the appropriate ODS PDF

statement and rerun the whole report again. If you had stored the output in an ODS document, you could simply open the ODS PDF destination and replay the report to get a printable version in a matter of seconds.

Another benefit to using ODS documents to replay entire reports is by varying ODS options, system options, macro variables, templates, and styles. All of these elements are re-evaluated when the report is replayed, so it is possible to affect various aspects of a report even if the original procedures are not executed again. Here is an example of changing the style on an HTML report when a report is replayed:

```
ods html file='barrettsblue.html' style=styles.barrettsblue;
proc document name=mydocument;
   replay;
run;
ods html close;
```

HTML output from replaying the report with the new style is shown in Display 4:



The CONTENTS Procedure

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Monday, January 02, 2012 08:07:18 PM	Observation Length	40
Last Modified	Monday, January 02, 2012 08:07:18 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Student Data		
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Engine/Host Dependent Information	
Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	101
Obs in First Data Page	19
Number of Data Set Repairs	0
Filename	C:\SASv9\sasgen\dev\mva-v940\sas_dvd\sio\dntnd\en\sashelp\class.sas7bdat
Release Created	9.0401B0
Host Created	NET_SRV

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

Display 4. HTML Output from Replaying a Report Using a New Style

While replaying an entire report does have its uses, it doesn't really show all that ODS DOCUMENT and PROC DOCUMENT are capable of. Let's move on to more advanced uses of the REPLAY statement.

REPLAYING PARTIAL REPORTS

In Output 1, we showed a listing of what is in a small document file. Each of the entries in that output had the following form:

```
\directory-1#x\directory-n#y\object#z
```

These items are called "document paths." Segments in a path are separated by a backslashes (\). Each segment in a path consists of a name (directory-1, directory-n, or object) and a sequence number (#x, #y, #z). The segment names can be the same at any one directory level, so the sequence numbers are used to allow each segment in a document to be uniquely addressable. When specifying a path, the sequence numbers are optional. However, keep in mind that if you leave the sequence number off and there are multiple items with the same name in that directory, the last one will be selected. These paths are very similar to Windows and UNIX file system paths. We will see later in this paper that paths and objects can be manipulated very much like files in a file system. For now, we will use the existing paths to replay components of a report.

We have already seen that a report can be replayed in its entirety by using PROC DOCUMENT's REPLAY statement, but it is also possible to replay part of a report by specifying the object's path. The first table shown in Output 1 is \Contents#1\DataSet#1\Attributes#1. This corresponds to the first table output by PROC CONTENTS. To replay just that one table, we can use the REPLAY statement as follows:

```
ods html file='replayone.html';
proc document name=mydocument;
   replay \Contents#1\DataSet#1\Attributes#1;
run;
ods html close;
```

The output for the code above is shown in Display 5:

The CONTENTS Procedure

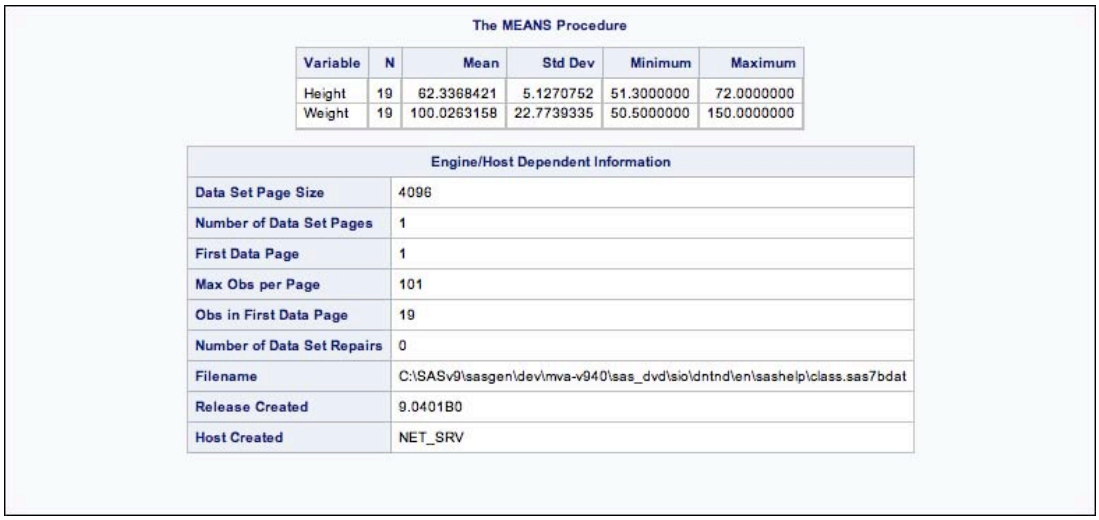
Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Monday, January 02, 2012 08:07:18 PM	Observation Length	40
Last Modified	Monday, January 02, 2012 08:07:18 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Student Data		
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Display 5. Output from Replaying Just One Table

As you can see, the report has only one table since that was all that was specified on the REPLAY statement. While this type of operation is similar to that of the ODS SELECT and ODS EXCLUDE statements, ODS DOCUMENT goes well beyond the capabilities of those statements. Those statements turn output objects on or off, but you cannot affect the order in which the procedure renders them. With the PROC DOCUMENT REPLAY statement, you can specify whatever order you wish. You can even specify more than one output object per REPLAY statement. Here is an example that replays the PROC MEANS output using the directory name and the EngineHost table from PROC CONTENTS:

```
ods html file='replaymult.html';
proc document name=mydocument;
   replay \Means#1, \Contents#1\DataSet#1\EngineHost#1;
run;
ods html close;
```

The output for the code above is shown in Display 6:



The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Height	19	62.3368421	5.1270752	51.3000000	72.0000000
Weight	19	100.0263158	22.7739335	50.5000000	150.0000000

Engine/Host Dependent Information	
Data Set Page Size	4096
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	101
Obs in First Data Page	19
Number of Data Set Repairs	0
Filename	C:\SASv9\sasgen\dev\mva-v940\sas_dvd\sio\dntnd\en\sashelp\class.sas7bdat
Release Created	9.0401B0
Host Created	NET_SRV

Display 6. Output from Replaying a Directory and a Table

When replaying directories, it is possible to limit the number of levels within those directories to be replayed. For example, if you only want to replay the objects within the first level of a directory, you would specify LEVELS=2 on the REPLAY statement (e.g., LEVELS=1 refers to the directories themselves, LEVELS=2 refers to the output objects in the level 1 directories), as shown below.

```
proc document name=mydocument;
   replay / levels=2;
run;
```

Since the only output object in the first two levels of the document is the Summary table of PROC MEANS, only that table will be created.

So far we have been explicitly specifying paths to replay, but there is a way to replay parts of a document based on WHERE clauses. This will be discussed in the following section.

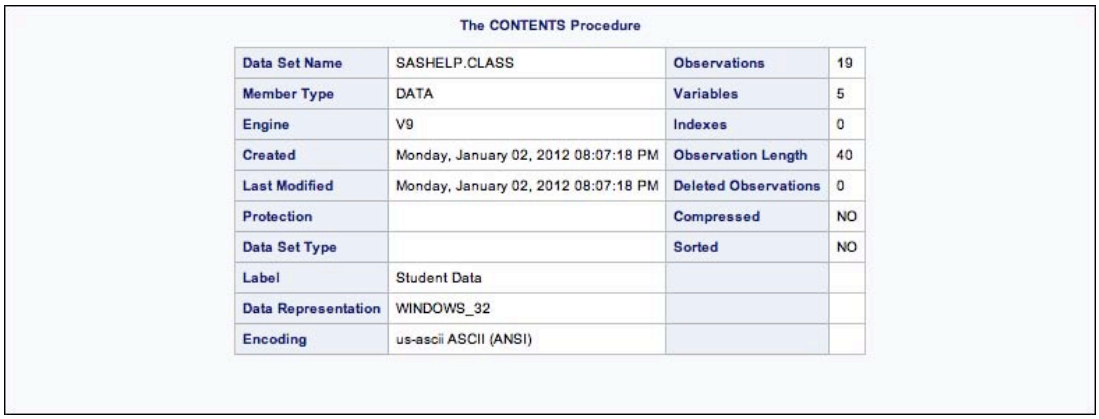
REPLAYING WITH WHERE CLAUSES

Replaying with explicit paths does give you a lot of power that isn't available in any other way in ODS, but it still feels a little constricting. However, using WHERE clauses to select output objects from a document opens up a whole new world of customized reports. Here is a simple example of using a WHERE clause to select output objects from a document. It selects any output object where _name_ is "Attributes":[2]

```
proc document name=mydocument;
   replay \ ( where=( _name_ = 'Attributes' ) );
run;
```

[2] The backslash after REPLAY selects the root element of the document. This applies the WHERE clause to the entire document.

The output for the code above is shown in Display 7:



The CONTENTS Procedure

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Monday, January 02, 2012 08:07:18 PM	Observation Length	40
Last Modified	Monday, January 02, 2012 08:07:18 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Student Data		
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Display 7. Output Where _name_ Is "Attributes"

The WHERE clauses are standard SAS WHERE clauses and can use any operators or functions available to WHERE clauses in other parts of SAS. In addition, there are many variables available to you in PROC DOCUMENT. Table 1 outlines all of the currently available variables:

Name	Description
name	Name of the item
path	Path of the item
label	Label of the item
labelpath	Labels of all entries in path
type	Type of the item
min	First observation number
max	Last observation number
obs	Current observation number
seqno	Sequence number of the item
cdate	Creation date of the item
mdate	Modification date of the item
ctime	Creation time of the item
mtime	Modification time of the item
cdatetime	Creation date-time of the item
mdatetime	Modification date-time of the item
variable	BY variable (directories); Column variable (output objects)

Table 1. Variables for Use in WHERE Clauses in the REPLAY Statement

You can display the values of these variables using either the Documents window (type ODSDOCUMENTS in the command bar, then right-click each object and choose **Properties**) or use the PROC DOCUMENT LIST statement. The following LIST statement will display the information about all of the output objects in a document.

```
ods listing;
proc document name=mydocument;
   list / details levels=all;
run;
ods listing close;
```

Here is the output from the LISTING destination:

```
Listing of: \Work.Mydocument\
Order by: Insertion

                                                  Size
  Obs    Path                          Type      in Bytes         Created
-------------------------------------------------------------------------------
       1 \Contents#1                   Dir                    03JAN2012:12:49:31
       2 \Contents#1\DataSet#1         Dir                    03JAN2012:12:49:31
       3 \Contents#1\DataSet#1\        Table             708  03JAN2012:12:49:31
         Attributes#1
       4 \Contents#1\DataSet#1\        Table             531  03JAN2012:12:49:31
         EngineHost#1
       5 \Contents#1\DataSet#1\        Table             309  03JAN2012:12:49:31
         Variables#1
       6 \Means#1                      Dir                    03JAN2012:12:49:31
       7 \Means#1\Summary#1            Table             266  03JAN2012:12:49:31

  Obs          Modified      Symbolic Link   Template        Label
-------------------------------------------------------------------------------
       1  03JAN2012:12:49:31                                The Contents
                                                            Procedure
       2  03JAN2012:12:49:31                                SASHELP.CLASS
       3  03JAN2012:12:49:31                Base.Contents.  Attributes
                                            Attributes
       4  03JAN2012:12:49:31                Base.Contents.  Engine/Host
                                            EngineHost      Information
       5  03JAN2012:12:49:31                Base.Contents.  Variables
                                            Variables
       6  03JAN2012:12:49:31                                The Means
                                                            Procedure
       7  03JAN2012:12:49:31                base.summary    Summary
                                                            statistics

  Obs    Page Break
-----------------------
       1
       2
       3 Before
       4
       5
       6
       7 Before
```

Output 2 Verbose Oootput from the PROC DOCUMENT LIST Statement

Most of the variables in Table 1 are self-explanatory, but *variable* is complex and powerful enough for further clarification. When used on a directory, a variable refers to a BY group variable and can be used to subset a report to a given BY variable value. When used on an output object, a variable refers to a column variable and can be used to subset observations in a table. Let's look at a couple of examples.

We'll start with our current document and subset a table based on values within the table. The first thing we need to do is to determine what columns are available in a table and what values exist in those columns. This can be done using the ODS OUTPUT destination and PROC CONTENTS. The code below puts the Variables table object that is created by PROC CONTENTS into a data set named *var*. We then use PROC PRINT and PROC CONTENTS to display the output data set and its variable information:

ODS Document From Scratch, continued

```
ods output variables=var;
proc contents data=sashelp.class;
run;

ods listing;
proc print data=var;
run;
proc contents data=var;
run;
ods listing close;
```

The output from the code above is shown in Output 3:

Obs	Member	Num	Variable	Type	Len	Pos
1	SASHELP.CLASS	3	Age	Num	8	0
2	SASHELP.CLASS	4	Height	Num	8	8
3	SASHELP.CLASS	1	Name	Char	8	24
4	SASHELP.CLASS	2	Sex	Char	1	32
5	SASHELP.CLASS	5	Weight	Num	8	16

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Label
5	Len	Num	8	3.	
1	Member	Char	256		
2	Num	Num	8	1.	#
6	Pos	Num	8		
4	Type	Char	4		
3	Variable	Char	32		

Output 3. PROC PRINT and PROC CONTENTS Output from the ODS OUTPUT Data Set

Let's use this information to subset the PROC CONTENTS Variables table object to display just the height and weight observations. We can verify from the information above that the name of the column is "Variable" and it is a character variable. Using that information, we can use the following code to get the result that we want:

```
proc document name=mydocument;
   replay \Contents#1\DataSet#1\Variables#1
             (where=(variable='Height' or variable='Weight'));
run;
```

The subset report is shown in Display 8:

The CONTENTS Procedure

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
4	Height	Num	8
5	Weight	Num	8

Display 8. An Output Object Created by Subsetting Data Using A WHERE Clause in the REPLAY Statement

To demonstrate the use of WHERE clauses and BY group variables, we will have to construct a document that contains BY groups. We will use PROC MEANS and group the data set by sex and age.

```
proc sort data=sashelp.class out=class;
   by sex age;
run;

ods document name=bygroups;

proc means data=class;
   by sex age;
   var height weight;
run;

ods document close;
```

We can then use a WHERE clause, as in the following code, to display only the tables in the report with sex equal to "F" and age greater than 13.

```
proc document name=bygroups;
   replay \ ( where=( sex='F' and age > 13 ) );
run;
```

The resulting report is shown in Display 9:

The MEANS Procedure

Sex=F Age=14

Variable	N	Mean	Std Dev	Minimum	Maximum
Height	2	63.5500000	1.0606602	62.8000000	64.3000000
Weight	2	96.2500000	8.8388348	90.0000000	102.5000000

Sex=F Age=15

Variable	N	Mean	Std Dev	Minimum	Maximum
Height	2	64.5000000	2.8284271	62.5000000	66.5000000
Weight	2	112.2500000	0.3535534	112.0000000	112.5000000

Display 9. A Report Subset by BY Group Variables in the REPLAY Statement

We have only shown simple uses of the WHERE clause feature. The use of WHERE clauses to subset reports and output objects is really only limited to your imagination and what your data will allow. Replaying reports is only one of the many things that you can do with ODS documents. We'll leave this topic behind and look at how to modify documents in the following sections.

NAVIGATING AND MODIFYING DOCUMENTS

Up to this point, we have always referred to document paths explicitly or simply as the root node (\). However, ODS documents can be traversed much like Windows or UNIX file systems. You can address directories and output objects within documents by using paths that are relative to the current location within the document. This method of moving around documents becomes a useful tool when you modify them. You may wonder why, if you can replay parts of a document in any random order and subset as needed, would you need to modify them? While replaying an ODS document does give you a large amount of control over a report, it does not allow you to modify the information that is attached to the output objects in the document or the overall structure of the document that, in turn, controls the generated table of contents. Those elements can be changed by modifying the document itself.

We have mentioned the fact that the structure of a document parallels computer file systems. PROC DOCUMENT gives you tools that also parallel Windows and UNIX commands to copy, delete, and move elements in a document.

The PROC DOCUMENT commands and their corresponding Windows and UNIX commands are shown in Table 2:

Operation	**PROC DOCUMENT**	**Windows**	**UNIX**
Display the path of the current directory	dir	chdir	pwd
Change the current directory to *path*	dir *path*	chdir *path*	cd *path*
List the contents of the current directory or given path	list <*path*>	dir <*path*>	ls <*path*>
Copy a path	copy *a* to *b*	copy *a b*	cp *a b*
Move a path	move *a* to *b*	move *a b*	mv *a b*
Create a new directory	make *path*	mkdir *path*	mkdir *path*
Create a symbolic link	link *a* to *b*	N/A	ln –s *a b*
Create a hard link	link *a* to *b* / hard	N/A	ln *a b*
Rename a path	rename *a* to *b*	rename *a b*	mv *a b*
Delete a path	delete *path*	del *path*	rm *path*
Current directory specifier	^	.	.
Parent directory specifier	^^	..	..

Table 2. Table of PROC DOCUMENT Commands and the Corresponding Windows and UNIX Commands

There are some other commands that have no parallel in the Windows and UNIX world. These commands have to do with document management as a whole and SAS-specific metadata on the output objects in a document. These commands are outlined in Table 3.

Operation	**Command**
List all documents in *library*	doc library=*library*
Open *document* for update	doc name=*document*
Close the current document	doc close
Delete *document*	delete *document*
Import a data set, grseg, or text file to *path*	import data|grseg|textfile=*name* to *path*
Create a new note at *path*	note *path* "*text*"
Set the label of *path*	setlabel *path* "*text*"
Set the *n*th line before the note of *path*	obbnote<*n*> *path* "*text*"
Set the *n*th line after the note of *path*	obanote<*n*> *path* "*text*"
Set the *n*th line of the title of *path*	obtitle<*n*> *path* "*text*"
Set the *n*th line of the subtitle of *path*	obstitle<*n*> *path* "*text*"
Set the *n*th line of the footnote of *path*	obfootn<*n*> *path* "*text*"
Control the page breaks of *path*	obpage *path* / <after> <delete>
Display the template code for *path*	obtempl *path*
Hide *path* from being replayed	hide *path*
Unhide *path* from being replayed	unhide *path*

Table 3. Table of Remaining PROC DOCUMENT Commands

Let's continue using our document with PROC CONTENTS and PROC MEANS output for some navigation and modification examples.

MOVING AND SHAKING

We will work through a sample PROC DOCUMENT session that uses most of the commands from above to demonstrate how to modify an ODS document prior to replaying the document in order to restructure a report. This example will involve navigating an ODS document, copying and moving output objects, as well as changing titles and notes. Keep in mind that during the course of this example, PROC DOCUMENT is an interactive procedure and none of the statements actually take effect until a RUN statement is invoked.

The first thing we'll do is to open our document using PROC DOCUMENT as we have done many times already:

```
proc document name=mydocument;
```

We want to create a new workspace for our modifications. We'll do this by creating a new directory (folder) with the MAKE statement.

```
make MyFolder;
```

We are going to work with the output created by PROC CONTENTS. Rather than giving fully qualified paths to the objects in the Contents output, we will change our current working directory using the DIR statement and point to the Contents output folder so that we can use relative paths:

```
dir \Contents#1\DataSet#1;
```

To verify that we are in the \Contents#1\DataSet#1 directory, we can use the DIR statement to print the current working directory or the LIST statement to show what is in the current directory. Let's do the latter just to verify that we have the objects that we want to work with:

```
list; run;
```

```
Listing of: \Work.Mydocument\Contents#1\DataSet#1
Order by: Insertion
Number of levels: 1

  Obs       Path                                                    Type
-------------------------------------------------------------------------
        1   Attributes#1                                            Table
        2   EngineHost#1                                            Table
        3   Variables#1                                             Table
```

As you can see from the listing above, we are now in the \Contents#1\DataSet#1 directory. Let's copy the Attributes and Variables output objects to our working folder:

```
copy Attributes, Variables to \MyFolder;
```

We also want to demonstrate a MOVE command, so let's change directories and go to the \Means#1 directory to move the Summary object. Rather than using a fully-qualified path, we know that we are two directories deep relative to the Means#1 directory. Let's go up two directories and then go down to the Means#1 directory to demonstrate relative path movements. In Windows and UNIX, you would use a double-dot (..) to refer to a parent directory. However, periods in SAS have a special meaning to the language processor, so double-dot could not be used. Instead, PROC DOCUMENT uses a double-caret (^^):

```
dir ^^\^^\Means#1;
```

We can now move an output object from the Means#1 directory to MyFolder.

```
move Summary to \MyFolder\Summary;
```

We are now done moving things into our working directory, so let's change our working directory to \MyFolder for one last demonstration. Then, we'll check our work.

```
dir \MyFolder;
```

Now that we are in \MyFolder, let's rename the Attributes object to Attrs to demonstrate the RENAME statement:

```
rename Attributes to Attrs;
```

We've done quite a few operations so far. Let's review and verify what we've done. We first created a directory called MyFolder at the top of the ODS document. We then copied the Attributes and Variables objects from the PROC CONTENTS output to our new folder called MyFolder. After that, we moved the Summary object from the PROC MEANS directory to our new folder. Finally, the Attributes object was renamed to Attrs. To verify that all of this took place, we can use the LIST statement:[3]

```
list \ / levels=all;
run;
```

The output from the LIST statement is shown here. As you can see, we have objects called Attrs, Variables, and Summary in MyFolder, as expected.

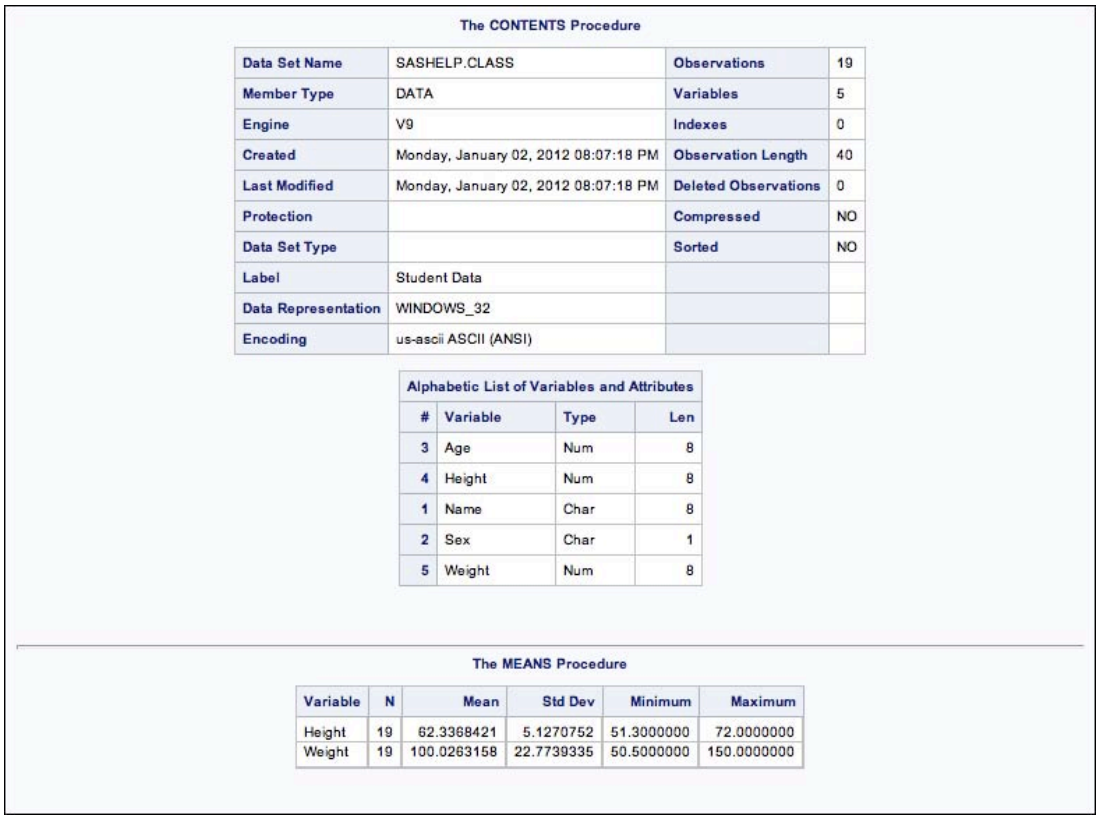
```
Listing of: \Work.Mydocument\
Order by: Insertion
Number of levels: All

  Obs    Path                                                                  Type
-------------------------------------------------------------------------------
       1 \Contents#1                                                          Dir
       2 \Contents#1\DataSet#1                                                Dir
       3 \Contents#1\DataSet#1\Attributes#1                                   Table
       4 \Contents#1\DataSet#1\EngineHost#1                                   Table
       5 \Contents#1\DataSet#1\Variables#1                                    Table
       6 \Means#1                                                             Dir
       7 \MyFolder#1                                                          Dir
       8 \MyFolder#1\Attrs#1                                                  Table
       9 \MyFolder#1\Variables#1                                              Table
      10 \MyFolder#1\Summary#1                                                Table
```

Now that we've collected all of the output objects that we want into MyFolder, we can replay that folder to see our new report.

```
ods html file='modify.html';
replay \MyFolder; run;
ods html close;
```

[3] Make sure that you include a RUN statement at the end. None of the statements are processed until the RUN is submitted.

The output from the REPLAY statement is shown in Display 10:



The CONTENTS Procedure

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Monday, January 02, 2012 08:07:18 PM	Observation Length	40
Last Modified	Monday, January 02, 2012 08:07:18 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Student Data		
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Height	19	62.3368421	5.1270752	51.3000000	72.0000000
Weight	19	100.0263158	22.7739335	50.5000000	150.0000000

Display 10. The Report Generated from a Modified ODS Document

The report is looking more like what we want, but there are still some enhancements that we can make to improve it. The page break before the Means table isn't necessary, and it would be nice to have more descriptive titles. Not to worry, these are things that can be handled with PROC DOCUMENT as well. First, let's set our current directory to \MyFolder to make sure we are in our workspace.

```
dir \MyFolder;
```

The page break can be removed using the OBPAGE statement.

```
obpage Summary / delete;
```

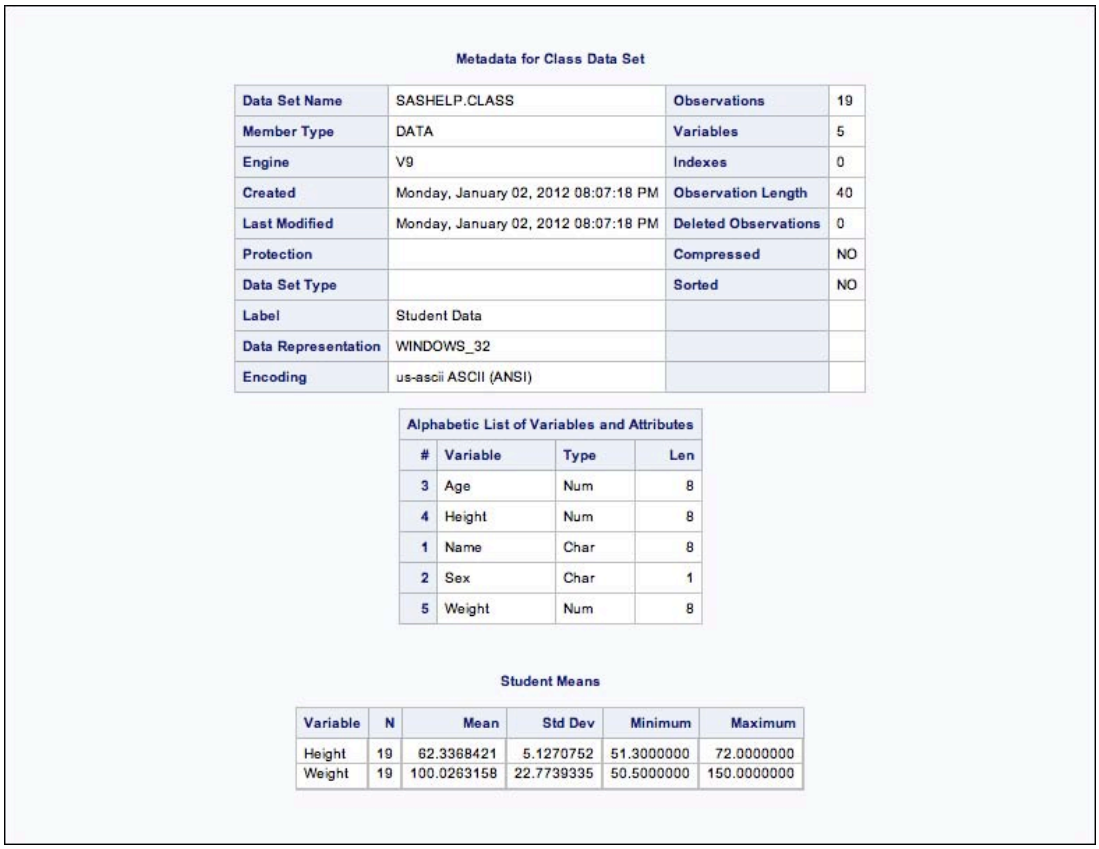
The titles and notes are controlled by the OBTITLE, OBSTITLE, OBANOTE, and OBBNOTE statements. Note that If any of these statements are specified without a text value, the title or note is deleted.

```
obstitle Attrs;
obbnote Attrs 'Metadata for Class Data Set';
obbnote Summary 'Student Means';
```

The first line of code above removes the procedure title. The second line adds a note before the Attrs table. The last line adds a note before the Summary table. With those modifications in place, we can replay MyFolder again. Since MyFolder is the current working directory, we'll use this as an opportunity to demonstrate the caret (^) operator, which corresponds to the current working directory much like the period operator (.) does in Windows and UNIX. Using the caret in this context means to replay the current directory:

```
ods html file='modify2.html';
replay ^; run;
ods html close;
```

The output from the REPLAY statement is shown in Display 11:



Metadata for Class Data Set

Data Set Name	SASHELP.CLASS	Observations	19
Member Type	DATA	Variables	5
Engine	V9	Indexes	0
Created	Monday, January 02, 2012 08:07:18 PM	Observation Length	40
Last Modified	Monday, January 02, 2012 08:07:18 PM	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label	Student Data		
Data Representation	WINDOWS_32		
Encoding	us-ascii ASCII (ANSI)		

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	Age	Num	8
4	Height	Num	8
1	Name	Char	8
2	Sex	Char	1
5	Weight	Num	8

Student Means

Variable	N	Mean	Std Dev	Minimum	Maximum
Height	19	62.3368421	5.1270752	51.3000000	72.0000000
Weight	19	100.0263158	22.7739335	50.5000000	150.0000000

Display 11. A Report Generated to Demonstrate Titles and Notes

If you aren't all that familiar with command-line navigation and file manipulation in Windows and UNIX, you may prefer to use the Documents window to modify your ODS documents.

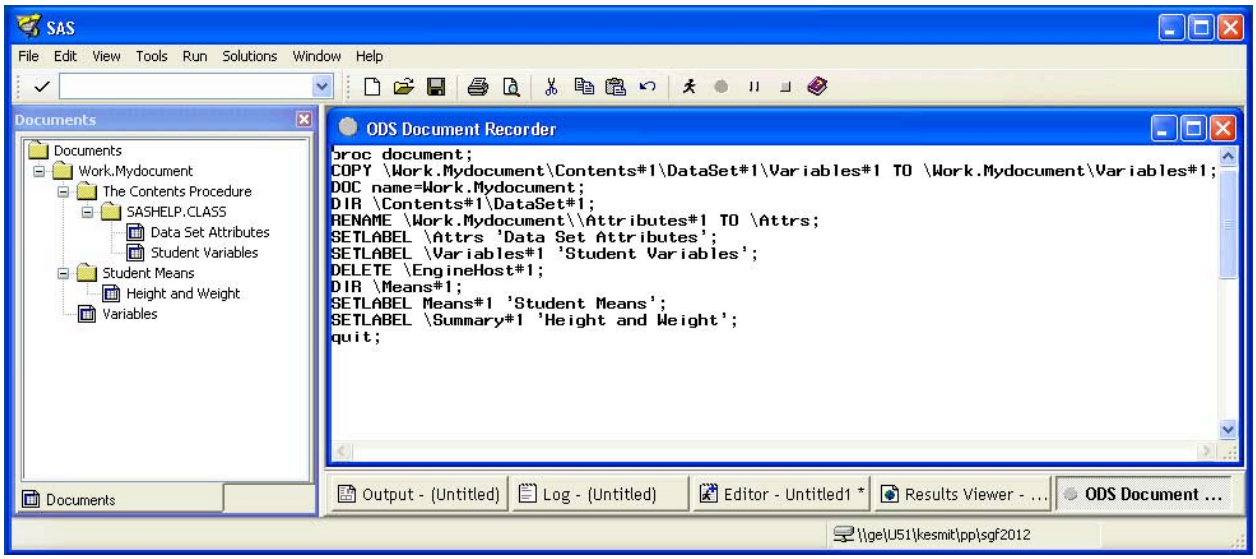
MODIFYING DOCUMENTS WITH THE DOCUMENTS WINDOW

Many of the document modification tools in PROC DOCUMENT are available from the Documents window as well. Modifying documents in this manner makes it similar to moving files and folders in Windows Explorer. This can be an easier way to get started with document modification. We discussed the Documents window earlier to browse a document. To invoke the Documents window, type ODSDOCUMENTS in the SAS command bar. You can then expand and collapse the tree view of the currently available documents.

Objects can be copied by dragging them to new locations. The standard cut, copy, and paste operations are available by keyboard shortcuts or the pop-up menu. Items can be deleted or renamed by using the Delete and Rename commands, respectively. The Rename command also enables you to change the label on an item. You can even create symbolic links with the Create Shortcut menu item. The only operations that are not available to the Documents window control the page breaks, titles, and notes.

While using the Documents window can be convenient, it doesn't work as well as code for reproducing modifications. To fill this gap, you can use the ODS Document Recorder that is available from the View menu. Any operations that are performed in the Documents window while the ODS Document Recorder window is open are recorded as PROC DOCUMENT statements. The code from this window can then be copied into your SAS programs for batch driven document modifications.

Display 12 shows the Documents window with modifications and the resulting code in the ODS Document Recorder window.



Display 12. The Documents Window with the ODS Document Recorder Window

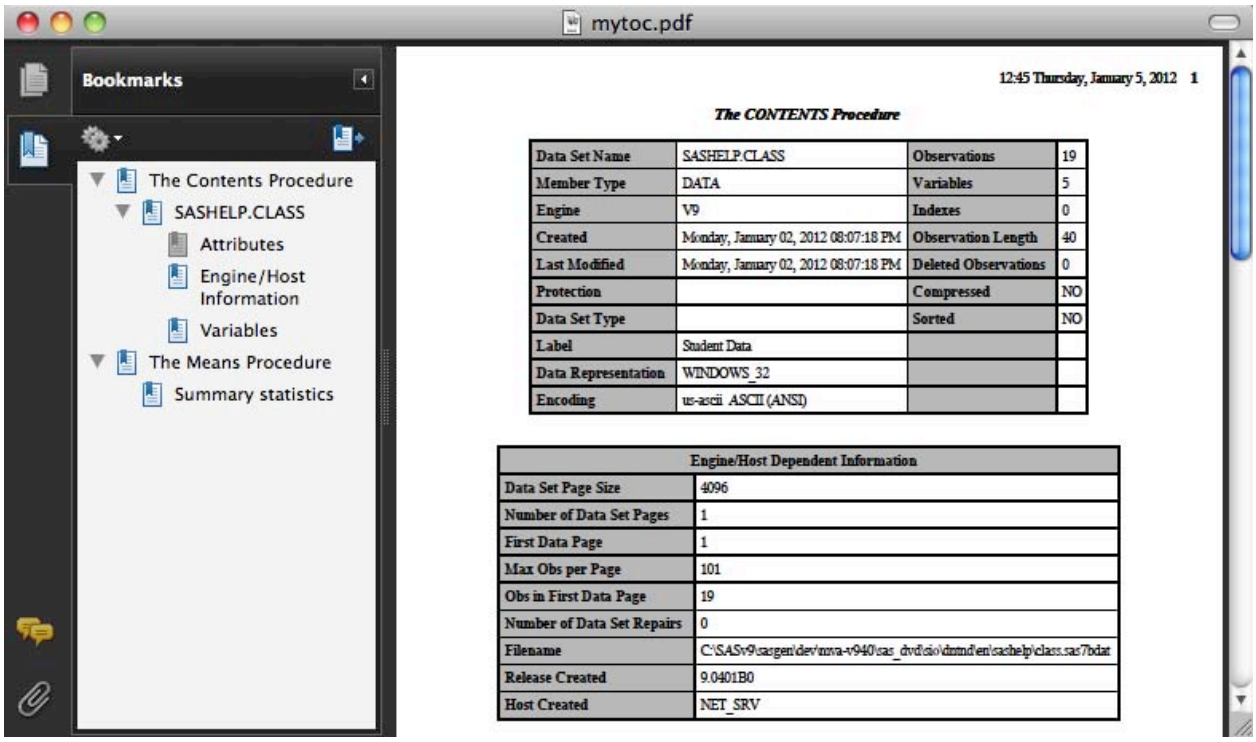
MODIFYING THE TABLE OF CONTENTS

One feature of ODS DOCUMENT that we haven't touched on yet is modifying the table of contents that is created by output destinations such as HTML, PDF, and RTF. Standard ODS statements give you minimal control over what shows up in the ODS table of contents. However, using ODS DOCUMENT, you can completely change the table of contents to fit your needs by changing the directory structure of the document.

Let's create a document with our familiar PROC CONTENTS and PROC MEANS output. This time we'll use the PDF destination so that we have only one output file:

```
ods document name=mytoc;
ods pdf file='mytoc.pdf';

proc contents data=sashelp.class;
run;

proc means data=sashelp.class;
run;

ods pdf close;
ods document close;
```

The PDF output is shown in Display 13:



Display 13. PDF Output Showing the Table of Contents

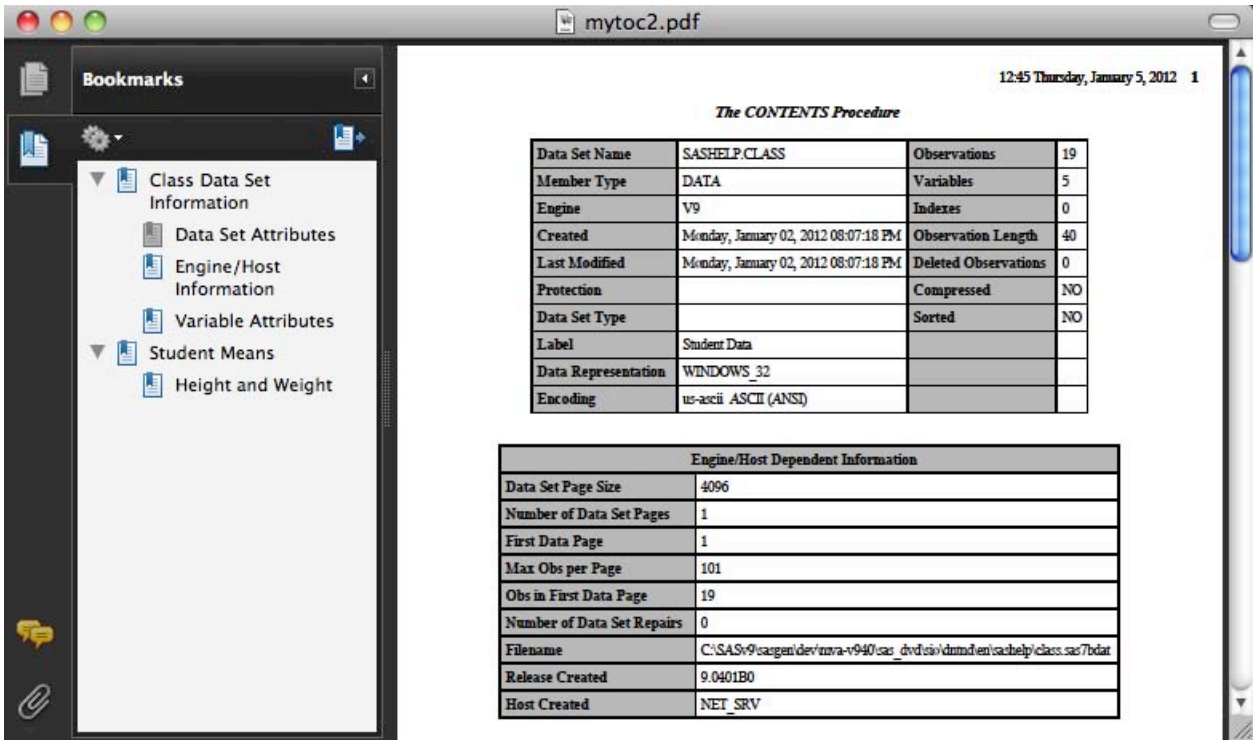
You may notice that the structure of the table of contents in the PDF is exactly the same as the directory structure of the ODS document. That will always be the case. So you may have guessed that by modifying the structure of the ODS document, you are also modifying the generated table of contents. Each directory in the ODS document corresponds to a folder in the table of contents, and each output object in the ODS document corresponds to a leaf node in the table of contents. If a label is set on a directory or output object, that label is used as the table of contents entry; otherwise, the name of the item is used.

The following code will move all of the PROC CONTENTS output objects up one level, delete the SASHELP.CLASS node, and change the labels to something more useful:

ODS Document From Scratch, continued

```
proc document name=mytoc;
   * Move PROC CONTENTS output up one level ;
   dir \Contents#1\DataSet#1;
   move Attributes, EngineHost, Variables to ^^;

   * Change labels on PROC CONTENTS output objects ;
   dir ^^;
   setlabel Attributes 'Data Set Attributes';
   setlabel Variables 'Variable Attributes';

   * Remove empty directory ;
   delete DataSet#1;

   * Set labels on top level directories ;
   dir \;
   setlabel Contents#1 'Class Data Set Information';
   setlabel Means#1 'Student Means';

   * Set label on PROC MEANS output object ;
   setlabel Means#1\Summary#1 'Height and Weight';

   ods pdf file='mytoc2.pdf';
   replay; run;
   ods pdf close;
run;
```

The PDF output from this code is shown in Display 14:



Display 14. PDF Output Showing a Modified Table of Contents

Of course, all of these modifications could also have been done using the Documents window and stored as code using the ODS Document Recorder.

There is another method of creating custom reports that doesn't require reorganizing the original document. This is useful when you want to create multiple variants of customized reports from a single document, or you just don't have the authority to change the original document. This type of customized report using symbolic links will be discussed in the following section.

AGGREGATE REPORTING USING SYMBOLIC LINKS

All of the reporting done so far in this paper has been replaying parts of a single document. However, it is possible to generate reports that span multiple documents. One way is to use the DOC NAME= statement to open documents then replay the paths from those documents in succession. This works fine, but the resulting table of contents structure may not be what is desired. Another more powerful way is to construct a new document that contains the directory structure corresponding to your desired table of contents, using symbolic links to point to output objects in external documents. A symbolic link is a shortcut to an object that is in another directory. This sounds harder than it is, so let's go to an example.

First, we need to generate multiple documents with various analyses. We'll stick with our PROC CONTENTS and PROC MEANS examples since we are familiar with the output paths, but this time we'll put the PROC CONTENTS output into one document and the PROC MEANS output into another document:

```
ods document name=contents;
proc contents data=sashelp.class;
run;
ods document close;

ods document name=means;
proc means data=sashelp.class;
   var height weight;
run;
ods document close;
```

We now have two documents containing information that we can use in our final report. We now need to create a new ODS document file from scratch. This is done by invoking PROC DOCUMENT with the name of our final report. Since we are creating a fresh report manually, we want to make sure that we aren't updating an old report. We ensure this by using Write mode.

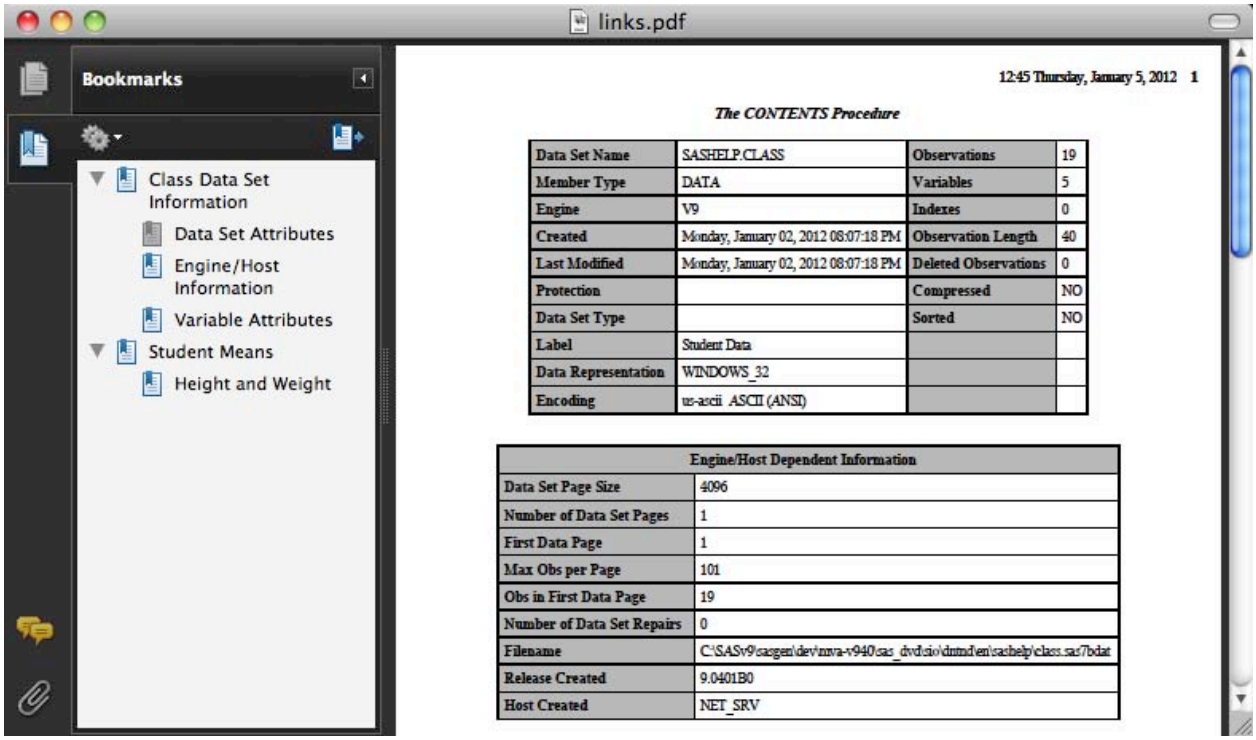
```
proc document name=links(write);
```

Let's mimic the structure of our last report so that we can show how to get the same results with two different methods. This will require creating two directories with the appropriate labels and linking to four output objects.

```
    * Create directory for PROC CONTENTS output ;
    make contents;
    setlabel contents 'Class Data Set Information';

    * Create directory for PROC MEANS output ;
    make means;
    setlabel means 'Student Means';

    * Link to Contents output objects ;
    link \work.contents\Contents#1\DataSet#1\Attributes to contents\attributes;
    link \work.contents\Contents#1\DataSet#1\EngineHost to contents\enginehost;
    link \work.contents\Contents#1\DataSet#1\Variables to contents\variables;

    * Set labels for table of contents ;
    setlabel contents\attributes 'Data Set Attributes';
    setlabel contents\enginehost 'Engine/Host Information';
    setlabel contents\variables 'Variable Attributes';

    * Link to Means output object and set label ;
    link \work.means\Means#1\Summary to means\summary;
    setlabel means\summary 'Height and Weight';

    * Replay the entire document ;
    ods pdf file='links.pdf';
    replay; run;
    ods pdf close;
run;
```

As you can see from Display 15, the PDF looks just like the PDF document from the previous example. The biggest difference is that in this case, we didn't have to modify the originally generated documents.

ODS Document From Scratch, continued



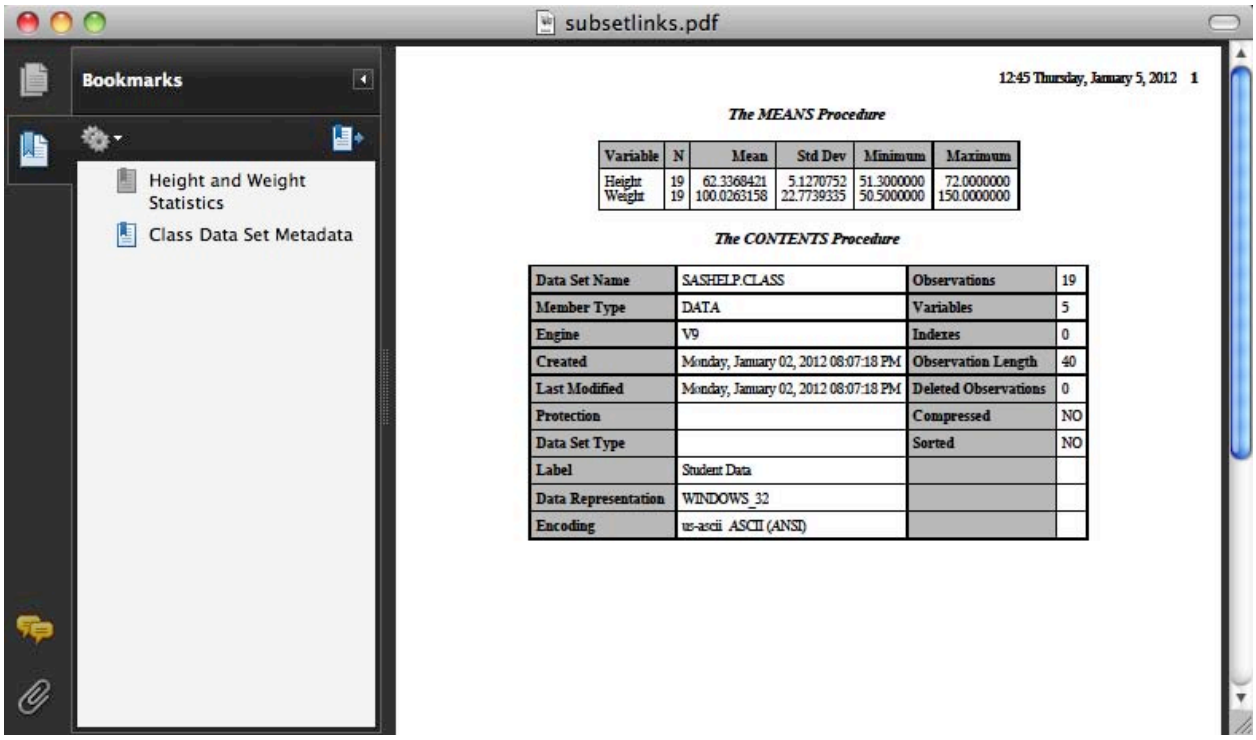
Display 15. PDF Output from an Aggregate Document That Uuses Symbolic Links

The biggest benefit to this method of creating reports is that you aren't limited to one view of the data. You can create as many aggregate reports as you wish that pull in objects from any number of ODS documents. This means that experts in any particular area can create rather generic documents containing their analyses, and anyone that is responsible for creating reports can create aggregate documents that point to the real data. The symbolic links in the document always point to the current version of the real data, so that every time the report is generated, it has the current data from all source documents.

To demonstrate a different view of the data from the previous report, here is another document that is created by using symbolic links. The document includes a subset of the original document's symbolic links and a different directory structure and labels:

```
proc document name=subsetlinks(write);
   * Create links at the top level ;
   link \work.means\Means#1\Summary to summary;
   link \work.contents\Contents#1\DataSet#1\Attributes to attributes;

   * Set labels;
   setlabel summary 'Height and Weight Statistics';
   setlabel attributes 'Class Data Set Metadata';

   * Replay report ;
   ods pdf file='subsetlinks.pdf';
   replay; run;
   ods pdf close;
run;
```

The resulting view of the data from this document is shown in Display 16:



Display 16. An Alternate PDF Report from an Aaggregate Ddocument Uusing Ssymbolic Llinks

CONCLUSION

As you can see from the examples in this paper, ODS Document and PROC DOCUMENT create many opportunities for customizing your reports that can't be done in any other way. They enable you to replay partial or entire reports without rerunning the procedures that generated the data. You can change ODS destinations, system options, and macro variable between replaying documents to give variants of the report. You can use WHERE clauses to programmatically select output that will be sent to the report. And finally, you can modify or create aggregate documents to construct entirely new table of contents structures and views of your data. This paper only scratched the surface of what is possible with ODS DOCUMENT in its simple examples. But with the tools presented here, you have the ability to customize your ODS reports beyond anything possible before.

RECOMMENDED READING

- *Base SAS® Procedures Guide*
- *SAS® Output Delivery System User's Guide*

CONTACT INFORMATION

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

Kevin D Smith
SAS Campus Drive
Cary, NC 27513

E-mail: Kevin.Smith@sas.com

EXAMPLE CODE

```
ods listing;
title;
footnote;

/* simple example, page 1 */
ods document name=mydocument;
ods html file='mydocument.html' style=styles.htmlblue;

proc contents data=sashelp.class;
run;

proc means data=sashelp.class;
   var height weight;
run;

ods html close;
ods document close;

/* LIST statement, page 3 */
proc document name=mydocument;
   list / levels=all;
run;

/* REPLAY statement, page 4 */
ods html file='replay.html' style=styles.htmlblue;
proc document name=mydocument;
   replay;
run;
ods html close;

/* style change, page 5 */
ods html file='barrettsblue.html' style=styles.barrettsblue;
proc document name=mydocument;
   replay;
run;
ods html close;

/* replay one document path, page 6 */
ods html file='replayone.html' style=styles.htmlblue;
proc document name=mydocument;
   replay \Contents#1\DataSet#1\Attributes#1;
run;
ods html close;

/* replay multiple document paths, page 6 */
ods html file='replaymult.html' style=styles.htmlblue;
proc document name=mydocument;
   replay \Means#1, \Contents#1\DataSet#1\EngineHost#1;
run;
ods html close;

/* replay levels=2, page 7 */
ods html file='replaylevels.html' style=styles.htmlblue;
proc document name=mydocument;
   replay / levels=2;
run;
ods html close;

/* simple where clause, page 7 */
ods html file='where.html' style=styles.htmlblue;
proc document name=mydocument;
```

ODS Document From Scratch, continued

```
   replay \ ( where=( _name_ = 'Attributes' ) );
run;
ods html close;

/* listing of all variables, page 8 */
proc document name=mydocument;
   list / details levels=all;
run;
quit;

/* inspect variables in proc contents variables table, page 9 */
ods listing close;
ods output variables=var;
proc contents data=sashelp.class;
run;
ods listing;
proc print data=var;
run;
ods select variables;
proc contents data=var;
run;

/* subset observations, page 10 */
ods html file='heightweight.html' style=styles.htmlblue;
proc document name=mydocument;
   replay \Contents#1\DataSet#1\Variables#1
              (where=(variable='Height' or variable='Weight'));
run;
ods html close;

/* subset by groups, page 10 and 11 */
proc sort data=sashelp.class out=class;
   by sex age;
run;

ods document name=bygroups;

proc means data=class;
   by sex age;
   var height weight;
run;

ods document close;

ods html file='bygroups.html' style=styles.htmlblue;
proc document name=bygroups;
   replay \ ( where=( sex='F' and age > 13 ) );
run;
ods html close;

/* modify a document, pages 12-15 */
proc document name=mydocument;

* Create a new folder ;
make MyFolder;

* Move to the \Contents#1\DataSet#1 directory ;
dir \Contents#1\DataSet#1;

* List the contents of the current directory ;
list;

* Copy the Attributes and Variables objects to MyFolder ;
```

ODS Document From Scratch, continued

```
copy Attributes, Variables to \MyFolder;

* Move to the Means directory ;
dir ^^\^^\Means#1;

* Move the Summary table to MyFolder ;
move Summary to \MyFolder\Summary;

* Move to MyFolder and rename the Attributes object ;
dir \MyFolder;
rename Attributes to Attrs;

* List the contents of the entire document ;
list \ / levels=all;

* None of the above statements run until we enter "run" ;
run;

* Replay MyFolder ;
ods html file='modify.html' style=styles.htmlblue;
replay \MyFolder; run;
ods html close;

* Move to MyFolder for more modifications ;
dir \MyFolder;

* Remove page breaks ;
obpage Summary / delete;

* Change titles and notes ;
obstitle Attrs;
obbnote Attrs 'Metadata for Class Data Set';
obbnote Summary 'Student Means';
run;

* Replay MyFolder again ;
ods html file='modify2.html' style=styles.htmlblue;
replay ^; run;
ods html close;

/* table of contents document */
ods document name=mytoc;
ods pdf file='mytoc.pdf';

proc contents data=sashelp.class;
run;

proc means data=sashelp.class;
   var height weight;
run;

ods pdf close;
ods document close;

/* modify table of contents, page 18 */
proc document name=mytoc label='My Report';
   * Move PROC CONTENTS output up one level ;
   dir \Contents#1\DataSet#1;
   move Attributes, EngineHost, Variables to ^^;

   * Change labels on PROC CONTENTS output objects ;
   dir ^^;
   setlabel Attributes 'Data Set Attributes';
```

ODS Document From Scratch, continued

```
   setlabel Variables 'Variable Attributes';

   * Remove empty directory ;
   delete DataSet#1;

   * Set labels on top level directories ;
   dir \;
   setlabel Contents#1 'Class Data Set Information';
   setlabel Means#1 'Student Means';

   * Set label on PROC MEANS output object ;
   setlabel Means#1\Summary#1 'Height and Weight';

   ods pdf file='mytoc2.pdf' startpage=never;
   replay; run;
   ods pdf close;
run;

/* symbolic link report, page 19 */
ods document name=contents;
proc contents data=sashelp.class;
run;
ods document close;

ods document name=means;
proc means data=sashelp.class;
   var height weight;
run;
ods document close;

proc document name=links(write);
   * Create directory for PROC CONTENTS output ;
   make contents;
   setlabel contents 'Class Data Set Information';

   * Create directory for PROC MEANS output ;
   make means;
   setlabel means 'Student Means';

   * Link to Contents output objects ;
   link \work.contents\Contents#1\DataSet#1\Attributes to contents\attributes;
   link \work.contents\Contents#1\DataSet#1\EngineHost to contents\enginehost;
   link \work.contents\Contents#1\DataSet#1\Variables to contents\variables;

   * Set labels for table of contents ;
   setlabel contents\attributes 'Data Set Attributes';
   setlabel contents\enginehost 'Engine/Host Information';
   setlabel contents\variables 'Variable Attributes';

   * Link to Means output object and set label ;
   link \work.means\Means#1\Summary to means\summary;
   setlabel means\summary 'Height and Weight';

   * Replay the entire document ;
   ods pdf file='links.pdf' startpage=never;
   replay; run;
   ods pdf close;
run;

/* another symbolic link view, page 20 */
proc document name=subsetlinks(write);
   * Create links at the top level ;
   link \work.means\Means#1\Summary to summary;
```

ODS Document From Scratch, continued

```
    link \work.contents\Contents#1\DataSet#1\Attributes to attributes;

    * Set labels;
    setlabel summary 'Height and Weight Statistics';
    setlabel attributes 'Class Data Set Metadata';

    * Replay report ;
    ods pdf file='subsetlinks.pdf' startpage=never;
    replay; run;
    ods pdf close;
run;
```