

# The Beginners Guide to ODS MARKUP: Don't Panic!

Eric Gebhart, SAS Institute Inc., Cary, NC

## ABSTRACT

The MARKUP destination is the most powerful Output Delivery System (ODS) destination. The ODS MARKUP statement can create a multitude of output file types from Comma Separated Values (CSV) to Compact HyperText Markup Language (cHTML), HTML 4.0, Extensible HyperText Markup Language (XHTML), LaTeX, and an endless variety of Extensible Markup Language (XML) flavors. It can also create Microsoft Excel output.

Learn how easy it is to use this destination, and learn how it works. Get the most out of this versatile ODS destination with minimum effort.

## INTRODUCTION

The plans to demolish ODS HTML were noted in SAS versions that preceded SAS 9.1. Subsequently, there were plans for ODS MARKUP; they just weren't posted in a public place. The plans weren't hidden at the bottom of a stairwell, underneath a file cabinet, or in a dark closet with a sign outside such as 'Beware of the Leopard'. All the plans were kept in my head. Then one day, the planets and stars aligned and, with a bang, ODS MARKUP emerged as a new and ever-expanding universe masquerading as an ODS destination.

This new universe consisted of all types of output in HTML, XML, CSV, LaTeX, and more. Immediately, ODS HTML became obsolete. To simplify and confuse matters, ODS MARKUP assumed the identity of ODS HTML, ODS CSV, ODS LaTeX, ODS Troff, ODS cHTML, and more. All these destinations were really just a part of the expanding universe of ODS MARKUP.

This new universe is a big place and exploring it can be a little scary. So, let me introduce you to the *Beginners Guide to ODS MARKUP*. Although this guide might not sell as well as the most famous book (*The Hitchhiker's Guide to the Galaxy*) that was printed by the great publishing corporations of Ursa Minor, the *Beginners Guide to ODS MARKUP* does take a clue from that famous tome, and it has the words *Don't Panic* in big friendly letters on its cover.

## THE BEGINNERS GUIDE TO ODS MARKUP

ODS MARKUP is more than it seems on the surface. That doesn't mean that you have to understand its intricacies and complexities to be able to take advantage of its capabilities. However, it's helpful to explore the unknown so that you know what is out there and how it works. You can explore the unknown through different levels of sophistication (for example, exploring a new universe by star gazing, looking at distant objects with a telescope, or launching an unmanned spacecraft) to obtain various degrees of information. A good approach is to build on your knowledge by starting "small", that is, start with the basics.

The first step in understanding the complexities of this new universe called ODS MARKUP is to understand markup languages. The *SAS 9.1 Output Delivery System User's Guide* says this about markup languages:

ODS Destinations:

...(extraneous information deleted)...

Markup languages produce SAS output that is formatted using one of many different markup languages such as HTML, XML, and LaTeX that you can access with a Web browser. SAS supplies many markup languages for you to use ranging from DocBook to Troff. You can specify a markup language that SAS supplies or create one of your own and store it as a user-defined markup language (Page 21).

Our soon-to-be best selling *Beginners Guide to ODS MARKUP* says this about markup languages:

A *markup* language is a textual way of adding context or formatting to data. A markup language can be as simple as CSV in which the only markup consists of commas that separate the data values. The most well-known markup language is HTML. The actual HTML consists of the tags that are enclosed by angle brackets (<>) and are placed around the content. These tags describe how to format the content. Troff is an old markup language that dates from the 1970's. Its syntax

consists of periods (.) and two-letter commands. LaTeX is another markup language and is used for publishing. Its syntax consists of back slashes (\), curly braces ({}), and somewhat long, descriptive, command names. XML is one of the most recent and more confusing markup languages. XML has very loose syntactic rules, and it allows you to make up your own language as you go along. This has the effect of causing data exchange to be very confusing and difficult because everyone has their own ideas about what their XML should look like.

To accommodate all these different types of textual output, it was necessary to make the ODS MARKUP destination programmable. ODS MARKUP cannot do anything without a program to tell it what to do. These programs are called *tagsets*.

The *SAS 9.1 Output Delivery System User's Guide* says this about tagsets:

A *tagset* is a type of template that defines how to generate markup language output from a SAS format. You can specify a tagset that creates markup language output from ODS. SAS provides tagset definitions for a variety of markup language output such as XML output, HTML output, XSL output, and more. You can modify any of the SAS tagsets or you can create your own. By supplying new tagset definitions, you can generate a wider variety of markup language output from SAS output (Page 551).

Our soon-to-be best selling *Beginners Guide to ODS MARKUP* says this about tagsets:

Tagsets were created by the somewhat warped mind of Eric Gebhart. Their purpose was to put ultimate control over ODS output into the hands of the ODS users, thereby freeing Eric to ride his bike and go on lots of vacations. The reality is that most people were quite content to let things be and have Eric exert all the control over their ODS output. The end result is that Eric spends even more time working on an ever-growing number of tagsets and their output. While most ODS users are content to use the tagsets that Eric creates, they have no desire to look at them.

The guide goes on to say this:

Tagsets are templates, as in templates for PROC TEMPLATE. This has not helped with their popularity. Despite the fact that tagsets are quite different from the table and style templates that are used by ODS, the reputation of the style templates has discouraged popular usage of tagsets.

For the purposes of this paper, it is most important to understand the relationship between a tagset and the ODS destination that it defines. Any destination that is defined by a tagset can be changed to do almost anything you can imagine. New possibilities are open to you just by knowing that "the sky is the limit" for those destinations.

For SAS 9.1, it's easier to list which ODS destinations are not tagsets than those that are tagsets. The destinations that are NOT tagsets are ODS PS (PostScript), PCL, PDF, and RTF. In the next major release of SAS, even RTF will be a tagset.

Many of the tagsets define ODS destinations that you can use all the time: HTML, cHTML, CSV, and LaTeX. There are a few others that look and behave as if they were any other ODS destination. This is the full list of tagsets that masquerade as true ODS destinations.

<u>HTML Destinations</u>	<u>WML Destinations</u>	<u>CSV Destinations</u>	<u>Publishing Destinations</u>
HTML	WML (Wireless Markup	CSV	Troff
HTML4	Language—an XML used	CSVALL	LaTeX
Msoffice2K	by phones and PDAs)		DocBook
PHTML	WMLOLIST		
HTMLCSS			
cHTML			
lmode			

Each of these destinations has a tagset that is associated with them. In most cases, the tagset name and the destination name match; the only exception is the HTML destination. By default, ODS HTML uses the HTML4 tagset. There are many more tagsets in addition to these. You can use PROC TEMPLATE to get a full list of the tagsets in SAS 9.1.

```
proc template;
  list tagsets;
run;
```

Listing of: SASHELP.TMPLMST  
Path Filter is: Tagsets  
Sort by: PATH/ASCENDING

Obs	Path	Type
1	Tagsets	Dir
2	Tagsets.Chtml	Tagset
3	Tagsets.Colorlatex	Tagset
4	Tagsets.Config_debug	Tagset
5	Tagsets.Csv	Tagset
6	Tagsets.Csvall	Tagset
7	Tagsets.Csvbyline	Tagset
8	Tagsets.Default	Tagset
9	Tagsets.Docbook	Tagset
10	Tagsets.Event_map	Tagset
11	Tagsets.ExcelXP	Tagset
12	Tagsets.Grandparent	Tagset
13	Tagsets.Graph_rtf	Tagset
14	Tagsets.Html4	Tagset
15	Tagsets.Htmlcss	Tagset
16	Tagsets.Htmlpanel	Tagset
17	Tagsets.Imode	Tagset
18	Tagsets.Latex	Tagset
19	Tagsets.MSOffice2k	Tagset
20	Tagsets.Mlatex	Tagset
21	Tagsets.Mvshtml	Tagset
22	Tagsets.Namedhtml	Tagset
23	Tagsets.Odsapp	Tagset
24	Tagsets.Odsgraph	Tagset
25	Tagsets.Odsstyle	Tagset
26	Tagsets.Odsxrpcs	Tagset
27	Tagsets.Phtml	Tagset
28	Tagsets.Pyx	Tagset
29	Tagsets.Rtf	Tagset
30	Tagsets.Rtf_sample	Tagset
31	Tagsets.Short_map	Tagset
32	Tagsets.Simplelatex	Tagset
33	Tagsets.Style_display	Tagset
34	Tagsets.Style_popup	Tagset
35	Tagsets.Supermap	Tagset
36	Tagsets.Tablesonlylatex	Tagset
37	Tagsets.Text_map	Tagset
38	Tagsets.Tpl_style_list	Tagset
39	Tagsets.Tpl_style_map	Tagset
40	Tagsets.Troff	Tagset
41	Tagsets.Wml	Tagset
42	Tagsets.Wmlolist	Tagset
43	Tagsets.Xbrl	Tagset
44	Tagsets.Xhtml	Tagset

Any of these tagsets can be used with the MARKUP destination just by specifying the tagset in the ODS MARKUP statement. For example,

```
ods markup tagset=ExcelXP file="test.xml";
```

Any tagset can be used as an ODS destination just by using its name as the destination name. For example,

```
ods tagsets.ExcelXP file="test.xml";
```

Our soon-to-be best selling *Beginners Guide to ODS MARKUP* says this about tagsets as destinations:

ODS MARKUP is only a name for the framework that allows users to define their own ODS destinations. The ODS destination is really the tagset. Therefore, it's recommended that the tagset name be used as the destination name in the ODS statement. The ODS statements will be more readable, and, as long as the names are unique, multiple tagset output destinations can be open at the same time.

## HTML DESTINATIONS

Of all the tagsets, the most popular generate HTML. Most people are familiar with the output of ODS HTML. The idea that ODS HTML is really a tagset within the ODS Markup framework might be a new concept. The ODS HTML destination uses the HTML4 tagset. HTML4 is the most complex HTML tagset. Its main purpose is to maintain backward compatibility and to fully exercise all the capabilities of ODS. There are other HTML tagsets that might meet your needs better than HTML4.

There is the cHTML tagset, also known as the Compact HTML tagset. cHTML is a subset of HTML and has no stylistic controls. cHTML has just the basics of HTML and was originally intended for phones and PDAs. Its compact size and simplicity made it perfect for small devices. Because of its simplicity, cHTML is popular among those who want black-and-white pages, and simple HTML. cHTML is also a good starting point for creating a custom-made HTML tagset of your own.

The Imode tagset provides HTML output based on the cHTML tagset, but without tables. This HTML output is used by several phone companies as the standard HTML for phones.

The PHTML (Plain HTML) tagset is another tagset that generates simple HTML. The PHTML tagset supports a very simple Cascading Style Sheet, and some, but not all, of the features that are supported by HTML4. Aside from the limited stylistic controls, the most obvious features that are missing are the style attributes of pre-HTML, post-HTML, pre-image, post-image, pre-text, and post-text.

The HTMLCSS tagset generates more complete HTML. The HTMLCSS tagset has everything that HTML4 has but without the worries of backward compatibility. It began as an idealistic attempt at pure W3C (World Wide Web Consortium)-compliant HTML.

The XHTML tagset is based on HTML4 and generates XHTML that is feature-identical to HTML4.

The following SAS program simultaneously creates output from all the various HTML tagsets.

```
ods imode file="imode.html";

ods chtml file="chtml.html";

ods phtml file="phtml.html";

ods htmlcss file="htmlcss.html";

ods html4 file="html4.html";

ods tagsets.xhtml file="xhtml.html";

proc reg data=sashelp.class;
  model Weight = Height Age;
run; quit;

ods _all_ close;
```

The output that's created with the Imode tagset is predictably ugly (Figure 1).

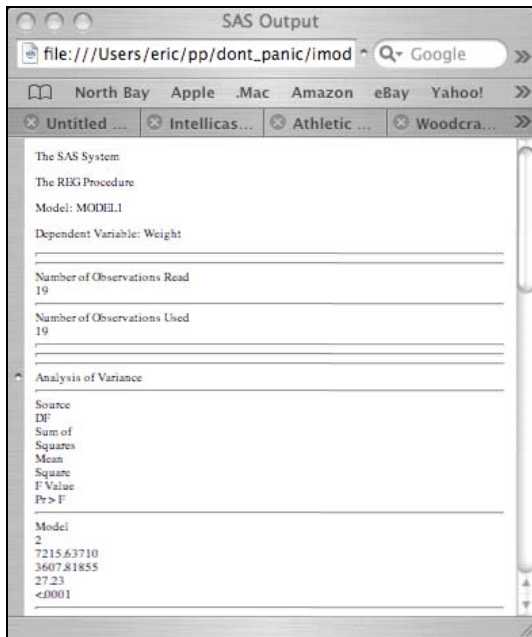


Figure 1. Output Created from the Imode Tagset

The output that's created with the cHTML tagset is better (Figure 2).

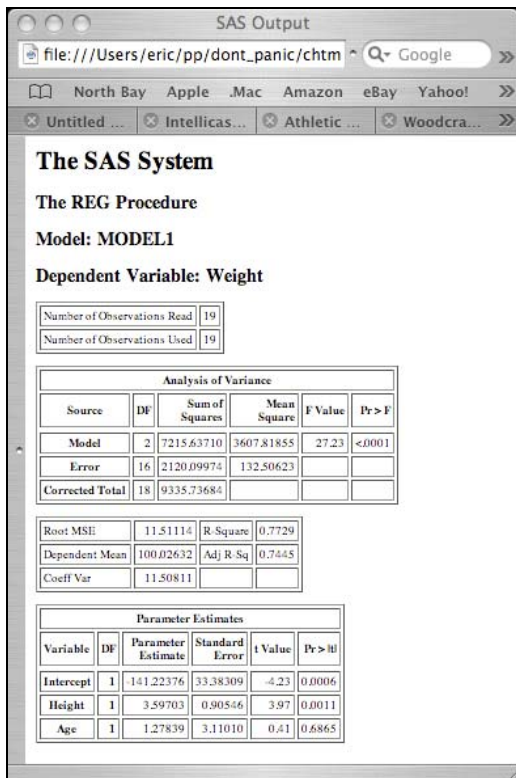


Figure 2. Output Created from the cHTML Tagset

The output that's created with the PHTML tagset has font and color but is still somewhat ugly (Figure 3).

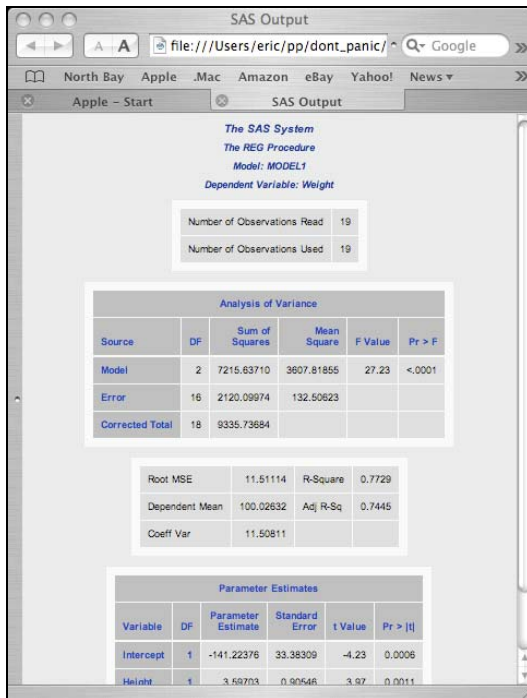


Figure 3. Output Created from the PHTML Tagset

The output that's created with the HTMLCSS tagset is much better (Figure 4).



Figure 4. Output Created from the HTMLCSS Tagset

The output that's created with the HTML4 and XHTML tagsets look identical (Figure 5).

The SAS System  
The REG Procedure  
Model: MODEL1  
Dependent Variable: Weight

Number of Observations Read	19
Number of Observations Used	19

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	7215.63710	3607.81855	27.23	<.0001
Error	16	2120.09974	132.50623		
Corrected Total	18	9335.73684			

Root MSE	11.51114	R-Square	0.7729
Dependent Mean	100.02632	Adj R-Sq	0.7445
Coeff Var	11.50811		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr >  t
Intercept	1	-141.22376	33.38309	-4.23	0.0006
Height	1	3.59703	0.90546	3.97	0.0011

Figure 5. Output Created from the HTML4 and XHTML Tagsets

## CSV DESTINATIONS

Comma separated values (CSV) format (and its variants) is still a very popular file format. There were originally three completely separate CSV tagsets: CSV, CSVALL, and CSVBYLINE. The latest version the CSV tagset incorporates all the functionality of these tagsets into a single tagset. The CSV tagset creates only tabular data in the output. CSVALL adds titles, footnotes, bylines, and notes. CSVBYLINE creates only the tabular data and bylines. All the tagsets are still available, but they all use the same tagset with different options set as needed to create the expected output. Here is an example of using the various CSV tagsets. The last CSV destination uses the ODS destination alias so that it does not clash with the first CSV destination that is already open. The last CSV destination also changes the delimiter to a semicolon.

```
ods csv file="csv.csv";

ods csvall file="csvall.csv";

ods tagsets.csvbyline file="csvbyline.csv";

ods csv(2) file="csvsemi.csv" options(Delimiter=';');

proc reg data=sashelp.class;
  model Weight = Height Age;
run; quit;

ods _all_ close;
```

The output from the first CSV statement looks like this:

```
"Number of Observations Read",19
"Number of Observations Used",19

"Analysis of Variance",,,,,
```

```

"Source","DF","Sum of Squares","Mean Square","F Value","Pr > F"
"Model",2,7215.63710,3607.81855,27.23,"<.0001"
"Error",16,2120.09974,132.50623,,
"Corrected Total",18,9335.73684,,,

"Root MSE",11.51114,"R-Square",0.7729
"Dependent Mean",100.02632,"Adj R-Sq",0.7445
"Coeff Var",11.50811,,

"Parameter Estimates",,,,,
"Variable","DF","Parameter Estimate","Standard Error","t Value","Pr > |t|"
"Intercept","1",-141.22376,33.38309,-4.23,0.0006
"Height","1",3.59703,0.90546,3.97,0.0011
"Age","1",1.27839,3.11010,0.41,0.6865

```

The Output from CSVALL looks very much the same, but with titles:

The SAS System

The REG Procedure

Model: MODEL1

Dependent Variable: Weight

"Number of Observations Read",19

"Number of Observations Used",19

"Analysis of Variance",,,,,

"Source","DF","Sum of Squares","Mean Square","F Value","Pr > F"

"Model",2,7215.63710,3607.81855,27.23,"<.0001"

"Error",16,2120.09974,132.50623,,

"Corrected Total",18,9335.73684,,,

"Root MSE",11.51114,"R-Square",0.7729

"Dependent Mean",100.02632,"Adj R-Sq",0.7445

"Coeff Var",11.50811,,

"Parameter Estimates",,,,,

"Variable","DF","Parameter Estimate","Standard Error","t Value","Pr > |t|"

"Intercept","1",-141.22376,33.38309,-4.23,0.0006

"Height","1",3.59703,0.90546,3.97,0.0011

"Age","1",1.27839,3.11010,0.41,0.6865

In this case, the CSVBYLINE will look no different than the CSV tagset does. If there had been BY-grouped data, the bylines would have shown up. The last CSV destination with the semicolon delimiters looks like this:

"Number of Observations Read";19

"Number of Observations Used";19

"Analysis of Variance";;;;;

"Source";"DF";"Sum of Squares";"Mean Square";"F Value";"Pr > F"

"Model";2;7215.63710;3607.81855;27.23;"<.0001"

"Error";16;2120.09974;132.50623;;

"Corrected Total";18;9335.73684;;;

"Root MSE";11.51114;"R-Square";0.7729

"Dependent Mean";100.02632;"Adj R-Sq";0.7445

"Coeff Var";11.50811;;

"Parameter Estimates";;;;;

"Variable";"DF";"Parameter Estimate";"Standard Error";"t Value";"Pr > |t|"

```
"Intercept";"1";-141.22376;33.38309;-4.23;0.0006
"Height";"1";3.59703;0.90546;3.97;0.0011
"Age";"1";1.27839;3.11010;0.41;0.6865
```

The CSV tagset is so versatile that it now has built-in Help. The number of tagsets with built-in Help is growing. The Help text is constantly growing as the tagset adds new features and capabilities.

The following ODS statement will cause the Help text to be printed to the log.

```
ods csv file="test.csv" options(doc='help');
```

Here is the Help text.

```
=====
The CSV Tagset Help Text

This Tagset/Destination creates output in comma separated value format.

Numbers, Currency, and percentages are correctly detected and show as numeric
values.
Dollar signs, commas, and percentages are stripped from numeric values by
default.

=====

These are the options supported by this tagset.

Sample usage:

ods csv options(doc='Quick');

ods csv options(currency_as_number='yes' percentage_as_number='yes'
delimiter=';');

Doc: No default value.
     Help: Displays introductory text and options.
     Quick: Displays available options.

Delimiter: Default Value ','
           Sets the delimiter for the values. Comma is the default. Semicolon is
           a popular setting for European sites.

currency_as_number: Default Value 'No'
                   If 'Yes', currency values will not be quoted.
                   The currency values are stripped of punctuation and currency symbols
                   so that they can be used as a number.

percentage_as_number: Default Value 'No'
                     If 'Yes', percentage values will not be quoted.
                     The percentages are stripped of punctuation and the percent sign (%)
                     so that they can be used as a number.

Currency_symbol: Default Value '$'
                 Used for detection of currency formats and for
                 removing those symbols so that Excel will like them.
                 Will be deprecated in a future release when it is no longer needed.

Decimal_separator: Default Value '.'
                  The character that's used for the decimal point.
                  Will be deprecated in a future release when it is no longer needed.
```

Thousands\_separator: Default Value ','  
The character that's used for indicating thousands in numeric values.  
Used for removing those symbols from numerics so that Excel will like them.  
Will be deprecated in a future release when it is no longer needed.

Bylines: Default Value: No  
If Yes, bylines will be printed.

Titles: Default Value: No  
If Yes, titles and footnotes will be printed.

Notes: Default Value: No  
If Yes, Note, Warning, Error, and Fatal notes will be printed.

Proc\_Titles: Default Value: No  
If Yes, titles generated by the procedures will be printed.

=====

Some of the more useful options are the options to turn currency and percentages into numbers. If they are turned on, the punctuation will be stripped from the currency values and the numbers will not be enclosed in quotation marks. That means that when Excel imports the CSV file, all the numbers will be imported as numbers instead of quoted strings. It is still problematic because the proper formats will still need to be applied in Excel.

Our soon-to-be best selling *Beginners Guide to ODS MARKUP* says this about the OPTIONS option:

The OPTIONS option was introduced in the ODS MARKUP statement in SAS 9.1.3. There was a recognized need for more dynamic control over the behavior of the tagsets. The options are really an arbitrary list of name value pairs. The names can be any valid name, and the values are any string that's enclosed in quotation marks. The options appear to the tagset as an associative array named \$OPTIONS. The fully capitalized names become the keys in the array. It is up to the tagset to look at the \$OPTIONS array and use the values as it sees fit. After more than a few options were added, it became obvious that a DOCUMENTATION option might be a good idea. For tagsets that have options, `options (doc='help')` will give documentation for all the options that are available. One of the more peculiar aspects of the \$OPTIONS array is that it's just like any other tagset variable, its values can be set, modified, and deleted within the tagset.

The OPTIONS option is only available in SAS 9.1.3, but there are ways around that. One way is to use the same methodology that the CSV tagsets use. Create a new tagset with inheritance that sets the options the way that you want them. This is the complete source code for the CSVALL tagset.

```
define tagset tagsets.csvall;  
  parent= tagsets.csv;  
  notes "This is the CSV with titles and bylines definition";  
  
  define event initialize;  
    set $options['BYLINES'] 'yes';  
    set $options['TITLES'] 'yes';  
    set $options['PROC_TITLES'] 'yes';  
    set $options['NOTES'] 'yes';  
    trigger set_options;  
    trigger documentation;  
    trigger compile_regexp;  
  end;  
  
end;
```

The key to knowing how to do this is just knowing that you need to copy the initialize event from the parent tagset, and that the options are kept in the \$OPTIONS array. Setting the values in the \$OPTIONS array is equivalent to putting those options in the ODS MARKUP statement. Add the SET \$OPTIONS statements that you need before the set\_options event is triggered, and you have a tagset that behaves exactly as you want it to.

Using inheritance is the simplest and most powerful way to tweak a tagset to behave exactly as you want.

Our soon-to-be best selling *Beginners Guide to ODS MARKUP* says this about tagset inheritance:

Tagset inheritance has purposely been kept simple. There are enough complications without making the actual behavior overly complicated as it is in other PROC TEMPLATE languages. Using tagset inheritance is usually done rather simply. Find the tagset event that prints or does whatever you want to change, copy the event(s), and paste them into a completely new tagset that uses the original tagset as its parent. Edit the new tagset as needed. The hard part of all this is figuring out what you need to change.

## EXCELXP DESTINATION

If you want your SAS output to be loaded into Excel, then the ExcelXP tagset is the way to go. This tagset writes SpreadsheetML XML (from Microsoft). The worst limitation of the XML is that it only works with Excel 2000 and later. This tagset is extremely flexible and has set the record for having the most options. The tagset itself is updated at least once a year, so it's recommended that you check for new versions on the ODS MARKUP Web site available at [support.sas.com/rnd/base/topics/odsmarkup/](http://support.sas.com/rnd/base/topics/odsmarkup/). Also, a presentation and examples from SUGI 30 are available at [support.sas.com/rnd/base/topics/odsmarkup/pandp.html](http://support.sas.com/rnd/base/topics/odsmarkup/pandp.html).

The ExcelXP tagset offers complete control over worksheets in a workbook. You can

- tell the tagset if it should automatically create a new worksheet for each table, page, BY group, procedure, or not at all
- manually control what is on each worksheet and name the worksheets, or let them be named automatically
- embed titles in the worksheet or in the print preview
- create special titles and footnotes just for printing
- control a multitude of printing options for each worksheet, including column and row repeat, scale, and orientation.

The ExcelXP tagset automatically detects currency, percentages, and numbers so they display properly in Excel. It even enables you to set your own Excel formats and formulas as needed. The tagset recognizes Excel formulas when they occur as data in a cell, turns on autofilters for any set of contiguous columns, and sets frozen panes so that the headers and row headers don't scroll off the screen when the worksheet is too large to fit.

Column widths and row heights are automatically calculated based on the font size and data; when that is not good enough, those values can be set by using options.

Automatically create a table of contents worksheet and a worksheet with an index of worksheets created in the workbook.

Here's what using the ExcelXP tagset looks like.

```
ods tagsets.excelxp file="Excelxp.xls";

proc reg data=sashelp.class;

    model Weight = Height Age;

run; quit;

ods _all_ close;
```

The output looks like this (Figure 6).

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	7215.6371	3607.81855	27.23	< .0001
Error	16	2120.09974	132.50623		
Corrected Total	18	9335.73684			

Figure 6. Output Created Using the ExcelXP Tagset

Addressing special needs through inheritance works just as well for the ExcelXP tagset as it does for the CSV tagset. The worst thing about it is that the ExcelXP tagset is so complex that it might take some effort to figure out how to make it do what you want. The following example creates an Excel tagset that makes all the header columns vertical. The trick is to add 'ss:rotate="90"' to the header styles <Alignment> definition.

This is the piece of tagset code that does just that. A search of the tagset for <Alignment> found the place to add it.

```
put ' ss:Vertical="Top" ss:Rotate="90" ' /if cmp(htmlclass, 'header');
```

A copy-and-paste later makes the new tagset look like this. Don't Panic! Just copy-and-paste; add the PUT statement between the beginning and the end of the Alignment tag and it's done!

```
proc template;

  define tagset tagsets.vert_excelxp;
  parent = tagsets.excelxp;

  define event xl_style_elements;
  delstream style_elements;
  open style_elements;

  set $headerString lowercase(htmlclass);
  do /if index ($headerString, 'header');
    put '<Alignment';
    set $align_tag "True";
    put ' ss:WrapText="1"';
    unset $headerString;
  done;

  /*-----eric-*/
  /*-- Make column headers vertical... --*/
  /*-----10Jan06-*/
  put ' ss:Vertical="Top" ss:Rotate="90" ' /if cmp(htmlclass, 'header');

  do /if $vjust;
  put '<Alignment' /if ^$align_tag;
  set $align_tag "True";
  put ' ss:Vertical=';
  put '"Center"' /if cmp($vjust, 'm');
  put '"Top"' /if cmp($vjust, 't');
```

```

        put '"Bottom"' /if cmp($vjust, 'b');
done;
unset $vjust;

do /if $just;
    put '<Alignment' /if ^$align_tag;
    set $align_tag "True";
    put ' ss:Horizontal=';
    put '"Center"' /if cmp($just, 'c');
    put '"Left"' /if cmp($just, 'l');
    put '"Right"' /if cmp($just, 'r');
    put '"Right"' /if cmp($just, 'd');

else /if contains(htmlclass, "SystemTitle") or
contains(htmlclass, "SystemFooter") or
cmp(htmlclass, "Byline");
do /if ^$just;
    do /if cmp($align, "center");
        put '<Alignment' /if ^$align_tag;
        set $align_tag "True";
        put ' ss:Horizontal="Center"';
    done;
done;
done;
unset $just;

putl '>' /if $align_tag;
unset $align_tag;

trigger write_all_borders;

trigger font_interior;

put '<Protection';
put ' ss:Protected="1"';
put ' />' NL;

flush;
close;
end;

end;

run;

```

Using the new tagset looks like this:

```

/*-----eric-*/
/*-- Column widths will be too wide because the tagset thinks the   --*/
/*-- headers are horizontal...                                         --*/
/*-----10Jan06-*/

ods tagsets.vert_excelxp file="test.xls"
    options(absolute_column_width="2,8,2,2,4,5"
            row_heights="30"
            );

```

```

proc print data=sashelp.class;
run;

ods _all_ close;

```

The output looks like this (Figure 7).

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84
10	John	M	12	59	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90
13	Louise	F	12	56.3	77
14	Mary	F	15	66.5	112
15	Philip	M	16	72	150
16	Robert	M	12	64.8	128
17	Ronald	M	15	67	133
18	Thomas	M	11	57.5	85
19	William	M	15	66.5	112

Figure 7. Output with Vertical Header Columns Created Using the ExcelXP Tagset

A slightly better method would be to create a new header style for vertical headers. We can't specify text rotation in the style, but we can have complete control over the orientation of the headers if we name the style `vertical_header` and look for that name in the tagset. Then, the usual header style can be used, and `vertical_header` can be set by the procedure for headers that should be vertical.

Here is the revised program. The tagset only adds rotation for the `vertical_header` style. The program creates a new style that has a `vertical_header` style that is identical to the header style. PROC PRINT specifies the `vertical_header` styles where you want them.

```

proc template;

  define tagset tagsets.vert_excelxp;
  parent = tagsets.excelxp;

  define event xl_style_elements;
  delstream style_elements;
  open style_elements;

  set $headerString lowercase(htmlclass);
  do /if index ($headerString, 'header');
  put '<Alignment';
  set $align_tag "True";
  put ' ss:WrapText="1"';

```

```

        unset $headerString;
    done;

/*-----eric-*/
/*-- Make vertical column headers vertical... --*/
/*-----10Jan06-*/
    put ' ss:Vertical="Top" ss:Rotate="90" ' /if cmp(htmlclass,
'vertical_header');

    do /if $vjust;
        put '<Alignment' /if ^$align_tag;
        set $align_tag "True";
        put ' ss:Vertical=';
        put '"Center"' /if cmp($vjust, 'm');
        put '"Top"' /if cmp($vjust, 't');
        put '"Bottom"' /if cmp($vjust, 'b');
    done;
    unset $vjust;

    do /if $just;
        put '<Alignment' /if ^$align_tag;
        set $align_tag "True";
        put ' ss:Horizontal=';
        put '"Center"' /if cmp($just, 'c');
        put '"Left"' /if cmp($just, 'l');
        put '"Right"' /if cmp($just, 'r');
        put '"Right"' /if cmp($just, 'd');

    else /if contains(htmlclass, "SystemTitle") or
        contains(htmlclass, "SystemFooter") or
        cmp(htmlclass, "Byline");
        do /if ^$just;
            do /if cmp($align, "center");
                put '<Alignment' /if ^$align_tag;
                set $align_tag "True";
                put ' ss:Horizontal="Center" ';
            done;
        done;
    done;
    unset $just;

    putl '>' /if $align_tag;
    unset $align_tag;

    trigger write_all_borders;

    trigger font_interior;

    put '<Protection';
    put ' ss:Protected="1"';
    put ' />' NL;

    flush;
    close;
end;

end;

run;

```

```

/*-----eric-*/
/*-- Column widths will be too wide because the tagset thinks the --*/
/*-- headers are horizontal... --*/
/*-----10Jan06-*/

proc template;
  define style styles.mystyle;
    parent=styles.default;

    style vertical_header from header /
      ;
  end;
run;

ods tagsets.vert_excelxp
  style=mystyle
  file="test.xls"
  options(absolute_column_width="4,8,4,3,4,5"
    row_heights="30"
  );

proc print data=sashelp.class;
  var name / style(header) = vertical_header;
  var age sex;
  var weight height / style(header) = vertical_header;
run;

ods _all_ close;

```

The output now looks like this (Figure 8).

Obs	Name	Age	Sex	Weight	Height
1	Alfred	14	M	113	69
2	Alice	13	F	84	56.5
3	Barbara	13	F	98	65.3
4	Carol	14	F	103	62.8
5	Henry	14	M	103	63.5
6	James	12	M	83	57.3
7	Jane	12	F	84.5	59.8
8	Janet	15	F	113	62.5
9	Jeffrey	13	M	84	62.5
10	John	12	M	99.5	59
11	Joyce	11	F	50.5	51.3
12	Judy	14	F	90	64.3
13	Louise	12	F	77	56.3
14	Mary	15	F	112	66.5
15	Philip	16	M	150	72
16	Robert	12	M	128	64.8
17	Ronald	15	M	133	67
18	Thomas	11	M	85	57.5
19	William	15	M	112	66.5

Figure 8. Output Created with Specified Vertical Header Columns Using the ExcelXP Tagset

Even without customization the ExcelXP tagset is the best way to get ODS output into Excel. The amazing number of options gives it the flexibility to accomplish almost any report.

## LaTeX DESTINATIONS

LaTeX is a very powerful markup language based on TeX. TeX has been around for decades. LaTeX is free and runs on virtually all platforms. The LaTeX processor compiles the LaTeX markup into PostScript, PDF, DVI, and other file formats. There are four LaTeX tagsets shipped with SAS: LaTeX, ColorLaTeX, SimpleLaTeX, and TablesOnlyLaTeX. The first two tagsets are fully featured destinations that are similar to HTML4, RTF, or PDF. The only difference between them is the addition of color. SimpleLaTeX is a more basic form of LaTeX that generates output that is suitable for inclusion in other LaTeX documents or books. TablesOnly LaTeX generates that same simple LaTeX but ignores system titles and footnotes, notes, bylines, and so on.

The biggest advantage of using LaTeX is that it is possible to create PDF output that is customized beyond anything currently possible with ODS PDF. For anyone writing a book or paper in LaTeX, the LaTeX tagsets are the best way to include SAS output.

The following SAS program simultaneously generates LaTeX output from each of the four tagsets. More information about this program and the LaTeX tagsets are available at [support.sas.com/rnd/base/topics/odsmarkup/latex.html](http://support.sas.com/rnd/base/topics/odsmarkup/latex.html).

```
/* Legacy LaTeX for ODS. See Figure 9.*/
ods tagsets.latex file="legacy.tex";

/* Legacy LaTeX with color for ODS. See Figure 10.*/
ods tagsets.colorlatex file="color.tex" stylesheet="sas.sty"(url="sas");

/* Simple LaTeX output that uses plain LaTeX tables. See Figure 11.*/
ods tagsets.simplelatex file="simple.tex" stylesheet="sas.sty"(url="sas");

/* Same as above, but only prints out tables (no titles, notes, etc.).*/
/* Also, prints each table to a separate file */
ods tagsets.tablesonlylatex file="tablesonly.tex" (notop nobot) newfile=table;

proc reg data=sashelp.class;
  model Weight = Height Age;
run; quit;

ods tagsets.latex close;
ods tagsets.colorlatex close;
ods tagsets.tablesonlylatex close;
ods tagsets.simplelatex close;

/* Run each document twice since the longtable package requires it */
x pdflatex legacy.tex;
x pdflatex legacy.tex;
x pdflatex color.tex;
x pdflatex color.tex;
x pdflatex simple.tex;
x pdflatex simple.tex;
```

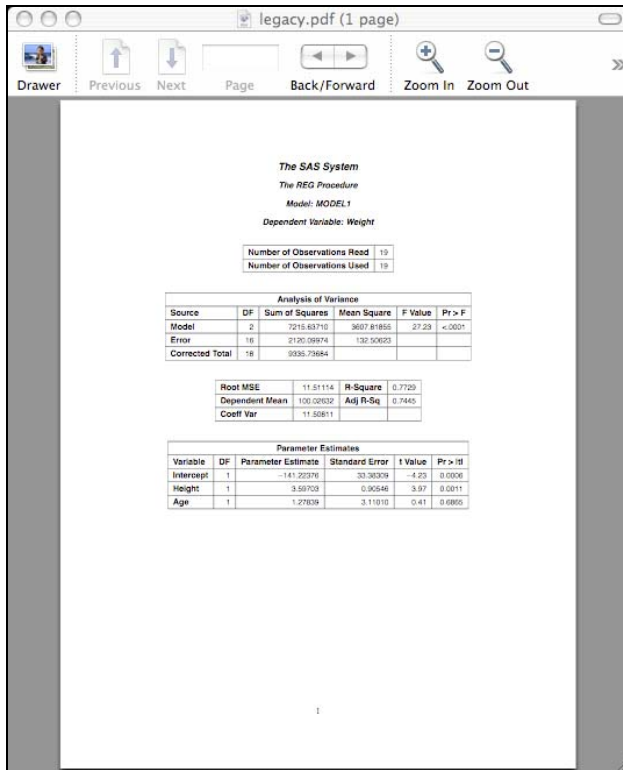


Figure 9. Output Created Using the LaTeX Tagset

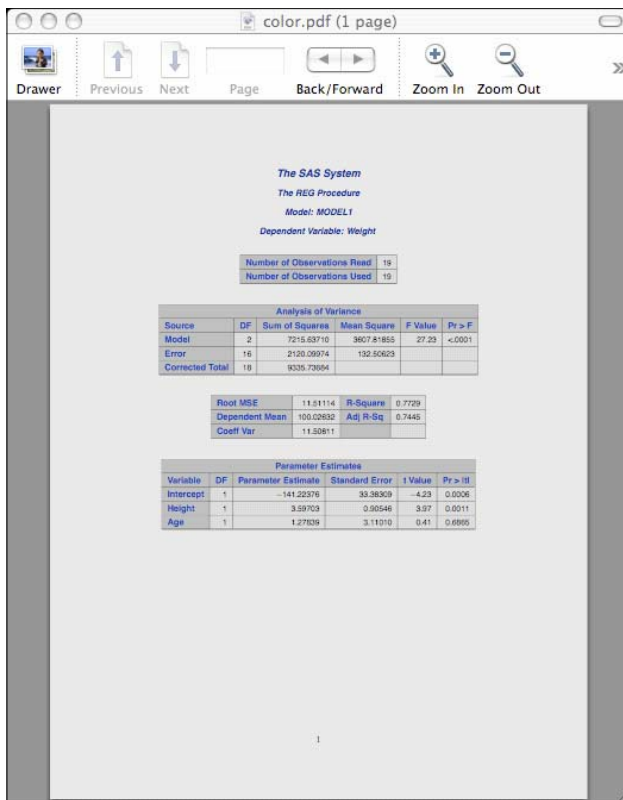


Figure 10. Output Created Using the ColorLaTeX Tagset

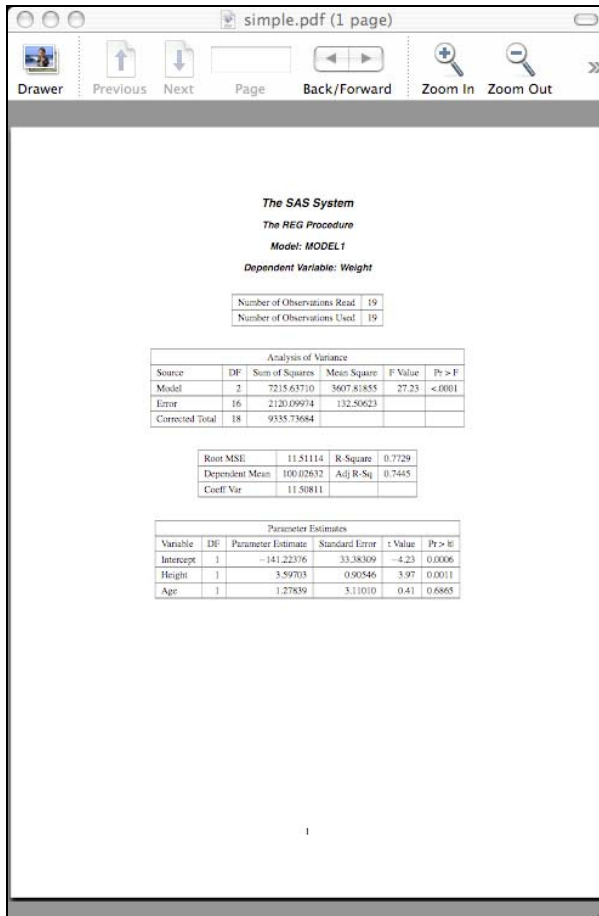


Figure 11. Output Created Using the SimpleLaTeX Tagset

It doesn't make any sense to compile the tablesonly.tex files because they are incomplete. By setting NOTOP and NOBOT, the files that are created are only LaTeX fragments. That makes them ideally suited for inclusion in existing LaTeX documents.

The following Web sites have all the information that you need in order to start using ODS LaTeX output to create great reports.

[www.ctan.org/](http://www.ctan.org/)

[www.latex-project.org/](http://www.latex-project.org/)

[www.tug.org/tetex/](http://www.tug.org/tetex/)

## CONCLUSION

The ODS MARKUP universe is still expanding and promises to continue expanding for a very long time. New destinations are constantly being created. The addition of measurement capabilities with the next major release of SAS has given ODS MARKUP the ability to supersede the current capabilities of ODS RTF with an RTF tagset. Those same features promise to make LaTeX an incredibly powerful destination. Understanding the ODS MARKUP universe will enable you to explore with confidence and discover possibilities that you could be using now and well into the future.

## **ACKNOWLEDGMENTS**

The author thanks Kathleen Walch and Josephine Pope for their work on this paper.

## **CONTACT INFORMATION**

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

Eric Gebhart  
SAS Institute Inc.  
Ste 2950  
1 PPG Place  
Pittsburgh, PA 15222  
Work Phone: (412) 227-0449  
Email: [Eric.Gebhart@sas.com](mailto:Eric.Gebhart@sas.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.