ODS Markup: Tagsets by Example

September 22, 2003

Preface

The SAS Output Delivery System's (ODS) markup destination appears to be much like all the other destination types that ODS is capable of. In reality it is quite different.

The other ODS destinations have a fixed output type. The RTF destination can only create RTF output. The printer destination can create postscript, PCL, and PDF. But ODS Markup can create any number of output types, all of which can be created or modified by anyone.

ODS Markup is able to do this because it is really a framework which uses a tagset to define what it should print to it's files. The other destinations, including the old HTML destination were hard coded and compiled as a part of the SAS executable.

A tagset can be described as a group of events or functions which use a programming syntax similar to other programming languages such as perl, python, shell, and datastep.

We can create and use as many different tagsets as we like. There are a number of tagsets that come standard as a part of SAS.

Because there are so many different tagsets, ODS Markup is rather chameleon-like. ODS HTML is really ODS Markup. So is the CSV destination, and CHTML, excelXP, LaTeX, Troff, etc. The only difference between these different ODS destinations is the tagset that is in use. They are all really ODS Markup, we just don't call it that.

When we use ODS Markup we are using a tagset. The tagset determines the type of output. Therefore, a tagset defines an output destination! This realization has far reaching implications. It means that we can change the html that ODS HTML generates. It means we can create a new HTML destination with your corporate style and headings. We can create a new XML destination to allow data interchange with a business partner or client. We can even create a simple flat file format that can simplify processing by other programs. The output can be modified and adapted in ways that could only be imagined until now. The possiblities are endless.

It stands to reason, that if we want to use ODS Markup to our best advantage, what we really want to do is use tagsets. That is what this book is going to explore. What are these tagsets and how do we use them? You will find that tagsets can simplify otherwise complex problems in a way that allows reuse and flexibility that would not be possible with out them.

ableofcontents

0.4pt1pt

Part I

Introduction

Chapter 1

Introduction

ODS markup and tagsets are the best way to create markup output from SAS. This chapter will talk about some of the reasons for learning tagsets. As well as what they are and the history behind them.

1.1 Why Learn Tagsets?

1.2 What are Tagsets?

1.3 A short History of Tagsets

1.4 Markup Languages

1.4.1 In the Beginning There Was Roff

1.4.2 Then there was LaTeX

1.4.3 SGML and friends

1.4.4 Other Formats

1.4.5 What it all means

Chapter 2

The basics

This chapter will cover the basics of using ODS Markup and the Template Procedure. ODS markup relies upon tagsets to define how it should work. Indicating which tagset to use is the only thing that differentiates the ODS Markup statement from any other ODS destination. A tagset is defined using the template procedure. The amount of template knowledge needed is not that great, but it is good to know the basics of using it.

2.1 Tagsets and Proc Template

2.1.1 The tagset directory

2.2 Using a tagset

2.3 Summary

Chapter 3

Tagsets: how do they work?

Tagsets are a hybrid of procedural and non-procedural, declaritive programming languages. A tagset is a collection of event definitions. Each definition can have procedural elements, but the events themselves are not procedural. That's why they are called events. Events do happen in some order, but with variation. They are like the events in your day. The alarm clock, brushing your teeth, eating breakfast, going to work, etc. They do have some predictability, but are not exactly the same all the time. This chapter will use a simple plain text tagset to show how tagsets work. As the tagset evolves the output created will reveal the fundamentals of tagset behavior.

3.1 Data and context in time

3.2 Event Requests

3.2.1 A few variables

3.3 Our First tagset

3.3.1 Define statement

3.3.2 The data event

3.3.3 The header event

3.4 Fleshing out the plain_text tagset

3.4.1 head, body, and foot

3.4.2 Titles

3.5 Summary

Chapter 4

Modifying Existing Tagsets

The most common tagset programming is done to modify the behavior of an existing tagset. Modifying a tagset is fairly easy to do because tagsets can inherit events from each other. It is also easy to identify which events need changing. This chapter will introduce a few new concepts while showing the steps needed to modify a tagset through inheritance.

4.1 Changing the delimiter for CSV files

4.1.1 Finding the Events

4.1.2 Simple If's

4.2 Making the changes

4.3 A better CSV tagset

4.3.1 Macro variables

4.3.2 The Initialize Event

4.3.3 The set statement

4.3.4 The New CSV Tagset

4.3.5 Tagset Alias

4.4 Summary

Chapter 5

The Path to Enlightenment

So far all of the examples have given away the events and variables needed. But the ability to find events, and their variables is required to be truly effective at programming tagsets. This chapter will explain several techniques that will illustrate which events may work for a solution, and the variables that are bundled in those events. In short, this chapter will show how to get tagsets to reveal themselves.

5.1 Finding Events

5.1.1 The Short_map Tagset

5.2 The Default_Event Tagset Attribute

5.2.1 The Events

5.2.2 The Default Event

5.2.3 summary

5.3 Finding Variables

5.4 The Putlog Statement

5.5 Define, Identify, Locate, Explore, and Solve.

5.5.1 Repeat as Necessary

5.6 Going step by step

5.6.1 Adding a Target to a URL

5.6.2 Define the Problem

5.6.3 Identify the event

5.6.4 Locate the Event

5.6.5 Explore the Data

5.6.6 Repeat. Identify, Locate, Explore

5.6.7 The solution

5.7 Summary

Chapter 6

File Redirection

Adding output to the all the files specified on the ODS statement requires that some events be redirected to those files. This chapter will show the differences between the various files and the events associated with them. It will also show how to use file redirection modify their contents.

6.1 The File Attribute

6.2 Identify

6.3 Locate and Explore

6.4 File interactions

6.5 Freedom of Choice

6.5.1 Explore

6.6 Summary

Part II

Technicalities

Chapter 7

The Tagset Attributes

A tagset can have several attributes. The attributes control the basic behavior of the tagset. Most importantly the translation of special characters that may interfere with the markup language being generated. Other attributes control output indention, breaking of lines, even the way the output should be displayed in the SAS output window. Setting up a tagset's attributes is the first step towards creating a new tagset.

7.1 Parent

7.2 Special Characters

7.2.1 Automatic Character Translation

7.3 Non Breaking Spaces

7.4 Split Characters

7.5 Indention

7.6 Stacked Columns

7.7 Image Formats

7.8 Output Type

7.9 Adding Measurements

7.10 Copyright Symbol

7.11 Trademark Symbol

7.12 Registered Trademark Symbol

7.13 Default Event

7.14 Embedded Stylesheet

7.15 Pure Style

7.16 lognote

7.17 Splitting Text

7.17.1 Breaktext Length

7.17.2 Breaktext Width

7.17.3 Breaktext Ratio

7.18 External Graph Instance

7.19 No Byte Order Mark

7.20 Hierarchical Data

7.21 Summary

Chapter 8

Creating Variables

The set statement has already been used in several examples. Set is the primary way to create and modify variables in tagsets. Before continuing it would be good to explain the different types of variables and the methods of creating them. This chapter is dedicated to explaining the different types of variables that can be created and used in tagsets.

8.1 String Variables

8.2 Lists

8.3 Dictionaries

8.4 Numeric Variables

8.5 Stream Variables

8.5.1 Stream Specific Statements

8.6 The Putvars Statment

8.7 Bringing it all together

8.8 Summary

Chapter 9

Procedural controls

The tagset's if syntax has already been used in previous examples. With SAS 8.2 and 9.0 if statements were kept to a few simple choices. This simple if statement is still the most efficient and most frequently used type of conditional programming that tagsets have. With SAS 9.1 if statments have been extended with where clauses and new statements to provide fully featured procedural controls. The examples in this chapter will explain the if syntax and the procedural control statements that complement them.

9.1 Simple If's

9.1.1 Built in tests

9.2 Where Clauses

9.3 Break

9.4 Breakif

9.5 Do blocks

9.6 Do While Loops

9.7 Iterating through Dictionaries

9.8 Bringing it all together

9.9 Summary

Chapter 10

Trigger Happy

ODS triggers, or requests, many events internally. It is also possible to trigger events from other events. This chapter is going to explain the trigger statement and how to use it.

10.1 Simple Triggers

10.2 Events with a state

10.3 summary

Chapter 11

Data Step Functions

Data step functions provide important capabilities to tagsets. Many solutions are dependent on string manipulation, formatting and number conversion. Without the functions, these solutions would be very difficult or impossible. This chapter will explain the usage of data step functions in tagsets.

11.1 Set and Put statements

11.2 The Eval Statement

11.2.1 number conversions

11.3 advanced usage and debugging

11.3.1 File I/O

11.3.2 Perl Regular Expressions

11.4 Summary

Part III

Intermediate Examples

Chapter 12

Styles and Tagsets, A perfect match

Understanding how to use ODS styles with tagsets can simplify many problems. Styles can help make a tagset more versatile and reusable. They can also help to make the output less verbose and more compartmentalized. This chapter will demonstrate the basics of using styles with tagsets. The best way to start is with a simple tagset designed specifically to show how styles work.

12.1 Starting Simple

12.1.1 Embedded styles

12.2 Styles of your own choosing

12.3 Getting the whole style

12.4 Summary

Chapter 13

Tagsets with Style

The fundamentals of tagsets have been covered, now it is time do something with them. This chapter will cover examples that specifically leverage ODS styles to accomplish their tasks.

13.1 A Problem with Table Rules

13.1.1 Define the problem

13.1.2 A Simple Solution

13.1.3 Identify and locate the event

13.1.4 A Simple Solution

13.1.5 Table Rules with style

13.1.6 The Better Solution

13.2 Everyone likes stripes

13.2.1 Defining the Problem

13.2.2 The HTML solution

13.2.3 The Code

13.2.4 The LaTeX solution

13.2.5 The Code

13.2.6 Summary

13.3 slidebar columns

13.3.1 Define the Problem

13.3.2 Identify and Locate

13.3.3 The Solution

13.3.4 Example Summary

13.4 Summary

Chapter 14

Tagsets with Streams

Streams add another level of versatility to tagsets. Streams allow for output to be saved away, and reused or just delayed. This is ideal for solving problems where the ODS event model does not match the shape of the target ouput. Streams are the most common solution to this problem. Streams can delay output, collect output from different events at different times, to be used later. This chapter will show how to use streams to solve problems that would otherwise be impossible.

14.1 Overly long Tables

14.1.1 Defining the Solution

14.1.2 Identify, Locate and Explore

14.1.3 The Solution

14.1.4 summary

14.2 Special Cases

14.3 One Better

14.3.1 The solution

14.4 A Tagset with Startpage

14.4.1 Identify, Locate and Explore

14.4.2 Defining the solution

14.4.3 Block and Unblock statments

14.4.4 A partial solution

14.4.5 The Solution

14.4.6 More Identify and Locate

14.4.7 Refining the Control

14.4.8 Almost There

14.4.9 The Final Solution

14.5 Summary

Part IV

Advanced Examples

Chapter 15

ODS Output for Website Integration

Many web sites have a framework that web pages must fit into before they will be published. Usually this means editing the html pages and pasting in some chunk of HTML from some files the website authors have provided. Usually there is one file to replace the top section of the web page and another to replace the bottom. The tagsets in this chapter will create HTML that is ready for integration with any website.

15.1 The Problem

15.2 Alternate Behavior for existing options

15.2.1 Explore

15.3 Reading an external file

15.4 The Solution

15.4.1 Initialization Timing

15.5 Summary

Chapter 16

Datastep Conversions

You may wonder, why would someone want to convert a datastep program to use tagsets? There are lots of reasons. Mostly it's for code reuse and flexibility. Datastep programs allow lots of flexibility but they are written for one purpose. If the same functionality is desired for a different set of data an entirely new datastep will have to be written. If the rendering is left to a tagset, then anyone can use that tagset with any form of data, and any procedure. Maintenance of the code is also simplified since both the tagset and the resulting SAS job wil be much simpler than the original datastep. Add the various powers of ODS to this new found flexibility and there is plenty of motivation for conversion. This chapter will show the process and rewards of converting datastep programs to tagsets.

16.1 Special Bylines

16.1.1 The DataStep Code

16.1.2 Breaking it down

16.1.3 The Style

16.1.4 Counting Observations

16.1.5 various problems

16.1.6 Modifying the Byline

16.1.7 A more flexible solution

16.2 Slidebars for HTML, PDF, and PS

16.2.1 breaking it down

16.2.2 The Style

16.2.3 The HTML Tagset

16.2.4 Dealing with the Report Procedure

16.2.5 The LaTeX Tagset

16.3 Summary

Chapter 17

Extended examples

This chapter will explore some more complex examples. The complexity comes mostly from the combination of multiple techniques. From a juggler's point of view, there are just more balls in the air. But a ball is still just a ball. Any given feature of these new tagsets is still just as simple as it ever was. This is one of the wonderful things about tagsets. It is easy to reuse and combine ideas from different tagsets.

17.1 Repeating Headers, and Mirrored Row headers

17.1.1 The Single Stream Solution

17.1.2 The Multiple Stream Solution

17.1.3 The Multiple List Solution

17.2 Automatic Panelling

17.2.1 An Extension of Start Page

17.2.2 The solution

17.3 HTML forms

17.3.1 Option Lists

17.3.2 Saving the Lists

17.3.3 Creating the Form

17.4 Summary

Chapter 18

A feature Rich Tagset

In this book we have created many tagsets that can each do something special. The tagsets have been carefully written to enable those features to be transparent to anyone who uses them. It is possible that all of these features be combined into one tagset. So that is what we are going to do.

18.1 Which features?

18.2 Lining up the inheritance

18.3 Copy and Paste

18.4 Macro Variables and Tagset Alias

18.5 Summary

Part V

Usage Notes and Caveat’s

Chapter 19

Special Cases, Procedures and Operating Systems’

The examples in this book have shown that there are some problems when it comes to the Report and Tabulate procedures. These problems are surmountable but they do cause some pain. This chapter is going to explain the specifics of these problems and how to work around them.

19.1 A Report Procedure problem

19.1.1 deferred data

19.2 The Tabulate and Report problem

19.2.1 The Table Head section

19.2.2 The Table column specifications

19.3 HTML on MVS with a PDSE

19.4 Summary

Chapter 20

Using LaTeX

Besides HTML and XML, LaTeX output is one of the most useful and versatile output destinations available. LaTeX is commonly used in publishing and is capable of creating several viewable formats, including PDF and Postscript. LaTeX supports color and various image formats. Sadly, this destination is largely overlooked. This chapter will discuss how to use it to advantage.

20.1 The LaTeX statement

20.1.1 Color Support

20.2 Compiling the LaTeX Output

20.2.1 The latex Command

20.2.2 The dvips Command

20.2.3 The pdflatex Command

20.3 Integrating LaTeX output into documents

20.3.1 The easy way

20.3.2 Using NewFile to advantage

20.3.3 The simple way

20.4 Image Formats and Graph

20.5 LaTeX in the different versions of SAS

20.5.1 SAS 8.2

20.5.2 SAS 9.0

20.5.3 SAS 9.1 and beyond.

20.6 Summary

Chapter 21

Using The HTML Tagsets

There are several HTML tagsets to choose from. Each generates a slightly different type of HTML. Each have their advantages. This chapter will explain the different HTML tagsets and the features they support.

21.1 The html statement

21.1.1 HTML4

21.1.2 PHTML

21.1.3 HTMLCSS

21.1.4 MSOffice2K

21.2 CHTML

21.3 Stylesheets

21.4 Javascript code

21.5 Scrolling Tables

21.6 Accessibility

21.7 HTML and Excel

21.8 Summary

Chapter 22

Using Markup output with Spreadsheets.

There are various ways to get SAS output into a spreadsheet. Each have their advantages and disadvantages. This chapter will go over the various tagsets that are good for importing into spreadsheet programs.

22.1 Using CSV

22.2 SYLK

22.3 HTML and Excel

22.3.1 Compact HTML

22.3.2 PHTML - Plain HTML

22.3.3 HTML for Microsoft Office 2000

22.4 DDE

22.5 Spreadsheet XML

22.6 Summary

Chapter 23

Using Tagsets with the Libname XML Engine

The libname XML Engine also uses tagsets. all of the output generated from the XML engine is done through a tagset. The libname engine's tagsets differ somewhat from the tagsets used by ODS Markup. This chapter will explain how to use tagsets with the libname engine and how they differ from those used by ODS.

23.1 XML Engine vs. ODS

23.2 Advantages of the XML engine

23.2.1 Control options

23.3 The Tagsets

23.4 Summary

Part VI

Appendices

Quick Reference Guide

.1 Useful tagsets

.2 Tagset attributes

.3 Event attributes

.4 Event Statements

.5 If Statements

Variables

.6 Event Variables

.6.1 508 Accessibility

.6.2 Data

.6.3 Event MetaData

.6.4 Data Formatting

.6.5 Title and Note Formatting

.6.6 Miscellaneous

.6.7 Graph

.6.8 ODS Statement

.6.9 Table

.6.10 URL

.6.11 XML Libname Engine

.7 Style Variables

.7.1 Borders

.7.2 Font

.7.3 Images

.7.4 Layout

.7.5 Miscellaneous

.7.6 Text

Extended Examples