# 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 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

# 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

# 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

## 2.2 Using a tagset

# 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

## 3.3 Our First tagset

## 3.4 Fleshing out the plain_text tagset

# Chapter 4

# Modifying Existing Tagsets

While you may find yourself creating a tagset from scratch, It is far more likely that all you'll want to do is modify the behavior of a tagset you already have. Modifying a tagset is fairly easy to do because tagsets can inherit events from each other. It's also much easier to identify which events need changing. In this chapter we 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

## 4.2 Making the changes

## 4.3 A better CSV tagset

## 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 allow you to understand how tagsets work, which events you may want to use, and the variables that are bundled in those events. This where you learn to get tagsets to reveal themselves.

## 5.1 Finding Events

## 5.2 Anatomy of the Short Map Tagset

## 5.3 The Tagset Attributes

## 5.4 Finding Variables

## 5.5 The Putlog Statement

## 5.6 Partial Enlightenment

# Part II

# Technicalities

# Chapter 6

# Creating Variables

We've used a set statement to create our delimiter variable in the CSV tagset. 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.

## 6.1 String Variables

## 6.2 Lists

## 6.3 Dictionaries

## 6.4 Numeric Variables

## 6.5 Stream Variables

## 6.6 The Putvars Statment

## 6.7 Bringing it all together

# Chapter 7

# Procedural controls

We've already been introduced to Tagset's if syntax. 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. This chapter will explain how those if statements work.

## 7.1 Simple If's

## 7.2 Where Clauses

## 7.3 Break

## 7.4 Breakif

## 7.5 Do blocks

## 7.6 Do While Loops

## 7.7 Iterating through Dictionaries

## 7.8 Bringing it all together

# Chapter 8

# Trigger Happy

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

## 8.1 Simple Triggers

## 8.2 Events with a state

## 8.3 summary

# Part III

# Intermediate Examples

# Chapter 9

# Styles and Tagsets, A perfect match

Understanding how to use 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. In this chapter we 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. From there we can move on to actual problems and their solutions.

## 9.1 Starting Simple

## 9.2 Styles of your own choosing

## 9.3 Getting the whole style

## 9.4 Summary

# Chapter 10

# Tagsets with Style

The fundamentals of tagsets are behind us, now it is time do something with them. In this chapter we will do examples that specifically leverage ODS styles to accomplish their tasks.

## 10.1 A Problem with Table Rules

## 10.2 Everyone likes stripes

## 10.3 slidebar columns

# Chapter 11

# 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 markup you are trying to create. In this chapter we will show how to use streams to solve problems that would otherwise be impossible.

## 11.1 Overly long Tables

## 11.2 Special Cases

## 11.3 One Better

## 11.4 A Tagset with Startpage

## 11.5 summary

# Part IV

# Advanced Examples

# Chapter 12

# ODS Output for Website Integration

Many web sites have a framework that web pages must fit into before they will be published on the web. Usually this means editing your html pages and pasting in some chunk of HTML from some files your website authors have provided. Usually there is one file to replace the top section of your HTML and another to replace the bottom. We can create a tagset that will create HTML that is ready for integration with your website. This chapter will show you how.

## 12.1 The Problem

## 12.2 Alternate Behavior for existing options

## 12.3 Reading an external file

## 12.4 The Solution

## 12.5 Summary

# Chapter 13

# 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 we have plenty of motivation for conversion. This chapter will show the process and rewards of converting datastep programs to tagsets.

## 13.1 Special Bylines

## 13.2 Slidebars for HTML, PDF, and PS

## 13.3 Summary

# Chapter 14

# extended examples

This chapter will explore some more complex examples. The complexity comes mostly from the combination of the many techniques we have learned so far. From a juggler's point of view, we just have 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.

## 14.1 Repeating Headers, and Mirrored Row headers

## 14.2 Automatic Panelling

# Chapter 15

# 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.

## 15.1 Which features?

## 15.2 Lining up the inheritance

## 15.3 Copy and Paste

## 15.4 Macro Variables and Tagset Alias

## 15.5 Summary

# Part V

# Usage Notes and Caveat’s

# Chapter 16

# Special Cases, The Report and Tabulate Procedures

We have seen, in our examples, 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.

## 16.1 A Report Procedure problem

## 16.2 The Tabulate and Report problem

# Chapter 17

# 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 your advantage.

## 17.1 The LaTeX statement

## 17.2 Compiling the LaTeX Output

## 17.3 Integrating LaTeX output into documents

## 17.4 LaTeX in the different versions of SAS

## 17.5 Summary

# Chapter 18

# 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.

## 18.1 The html statement

## 18.2 CHTML

## 18.3 Stylesheets

## 18.4 Javascript code

## 18.5 Scrolling Tables

## 18.6 HTML and Excel

# Chapter 19

# 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 good for importing into spreadsheet programs.

## 19.1 Using CSV

## 19.2 SYLK

## 19.3 HTML and Excel

## 19.4 DDE

## 19.5 Spreadsheet XML