

Lions, Tigers and Documentation???

Brett Peppe; Alan Mann; John Peppe; Chris Magill
Cedalion Risk Management, LLC

Abstract

Historically, relegated to the programmers “*I’ll get around to it bucket*” - the quality, legibility and style of SAS documentation varies considerably in both inherited codebase and ongoing development. Near term, the large body of legacy SAS programs will combine with accelerated knowledge worker turnover exposing inline and project documentation as an unanticipated cost center and risk surface. Put differently, SAS codebases can be deep, programmers are expensive and documentation matters now more than ever. This voluntary regime is being gradually supplanted by risk aware organizational management, guidance by standards organizations and harsher prescriptive regulations imposed by governmental bodies.

Introduction

For years after the appearance of SAS in the 1970’s individual programmers often wrote, ran and maintained their own code to create analytical work products for organizational consumers. Although, large SAS projects with sizable teams of programmers certainly existed (U.S. Census Bureau, U.S. Department of Agriculture, etc.) the majority of SAS programs were written by small teams or even individual programmers. Often these programs were run to generate reports on an as needed basis. Many users of SAS were not computer scientists or programmers by academic or professional training rather they were statisticians, engineers and others who had learned to use the best statistical and analytical platform available. This phenomena contributes to variation in the quality, legibility and style of SAS documentation. SAS programs were written by experts for experts – In this era, quality, legible, standardized inline documentation was most often an aid to oneself or courtesy to those who might follow; at most it was a best practice imposed by management or contract requirement. Similarly, the style and project documentation varied widely organization to organization.

“If I had asked what people wanted, they would have said faster horses.” Henry Ford

With time, the value of analytical work products exploded across entire industries and the world with the era of big data, SAS programs became incorporated into routine business processes. No longer was SAS simply a platform for specialized, often one off analysis, SAS programs became automated processes integral to day-to-day business operations. As the value of analytical work products exploded; so, to did the number of codebases, there complexity and the number of knowledge workers involved in the software lifecycle.

***“The thing's hollow it goes on forever and oh my God! it's full of stars!” David Bowman, 2001:
A Space Odyssey***

Gradual changes to career patterns through the 1980's and 1990's and the emergence of SAS development head-shops also reduced the likelihood that any given SAS program would be maintained by the same programmer across its entire lifecycle. Years passed and rate at which knowledge workers cycled increased, those who remained and often bewildered newcomers were tasked with maintaining legacy programs, with preserving the availability of work products, with migrating legacy functionality into new generations of business systems, regardless of whether they were the original author. Productivity and availability have ruled the waterfall and agile eras of software development. Priorities that have often caused documentation to languish as an afterthought.

Documentation Issues in a Nutshell

The Basics

- **What** does this program accomplish?
- **When** is this program run?
- **Where** are the inputs, processing and work product outputs?
- **Who** created this program. Who is maintaining it?
- **Why** was this program created the first place, the business justification.

The Problem

- SAS is both analytical frame and Toolset for **Knowledge workers**
- Knowledge workers are **Subject Matter experts (SME)** using a **tool**
- SAS Base coders are no longer being trained
- Many SAS programs answered a single business question **Once**
- But many SAS programs are now effectively in **Production**
- SAS program maintenance has often been deferred to cheaper contract staff

Technical Debt is Real

- The Fortune 500 are running SAS9 in **Production**

- Enormous Technical Debt vs High Business value
- The Meme: “If it isn’t Broken, Don’t Fix”
- Business is comfortable w/ existing work products
- Fear of (yet another) requirements, testing and acting cycle

The Road to Technical Debt

1. **Adhocs** - a SME is asked to answer a business question. Once. *Success?*
2. **Modhocs** - the same SME runs a parameter version version of (1) as needed, often weekly or Monthly. *Great Job!*
3. **Production** - the Organization runs (2) programmatically under Job Control. *Where did my report go?*

Operational Debt is Real

- Shrinking CapEx dollars
- Pressure on external Contractor budgets
- Increased organizational focus on Efficiency
- Emergent Regulatory Threat
- Unsure of baseline requirements and how they may have evolved
- H1B/Contract talent pool is drying up

Why you Need it

Corporations that license SAS often have hundreds or even thousands of important SAS programs scattered across their business units created by embedded knowledge workers and contractors. Because of changes in regulatory, risk aware organizational management and information security mandates, many SAS programs could be classified as critical business processes. They are now required at the minimum to have high level documentation of information flows. With audit and risk management and information security mandates tightening; quality, consistent inline and project documentation is no longer optional for:

- Yourself, team members or your management

- Legacy **code maintenance**
- Retention of **tribal knowledge**
- Migration planning to new release or products
- Regulatory compliance
- Audit compliance

Regulatory Threat

The Whirlpool is expanding

- Statutory Law (ex: CCPA, GDPR, etc.)
- Regulatory Law(ex: SOX, HIPPA, etc.)
- Standards (ex: NIST, ISO, COBIT, AICPA, BIS, etc.)
- Industry Best Practices (ex: PCI/DSS, etc.)
- Executive Orders(...)

is *emergent*

- Each supports separate vertical industries
- Have specific legal contexts
- Do not agree of Form of documentation
- Vary across jurisdictions
- Will likely consolidate / merge over time

\

The Feds are Coming, the Feds are Coming ...

Large organizations have a lot of investment in important SAS programs. Nearly all of them sell services to the Federal government.

[Executive Order](#) which was signed on May 11 2024 mandated changes on all government agencies evaluate risk.

NIST Cybersecurity Framework Version 1.1 to NIST Special Publication 800-53, Revision 5, Security and Privacy Controls for Information Systems and Organizations			
Function	Category	Subcategory	NIST SP 800-53, Revision 5 Control
	Asset Management (ID.AM): The data, personnel, devices, systems, and facilities that enable the organization to achieve business purposes are identified and managed consistent with their relative importance to organizational objectives and the organization's risk strategy.	ID.AM-1: Physical devices and systems within the organization are inventoried	CM-8, PM-5
		ID.AM-2: Software platforms and applications within the organization are inventoried	CM-8
		ID.AM-3: Organizational communication and data flows are mapped	AC-4, CA-3, CA-9, PL-8, SA-17
		ID.AM-4: External information systems are catalogued	AC-20, PM-5, SA-9
		ID.AM-5: Resources (e.g., hardware, devices, data, time, personnel, and software) are prioritized based on their classification, criticality, and business value	CP-2, RA-2, RA-9, SA-20, SC-6
		ID.AM-6: Cybersecurity roles and responsibilities for the entire workforce and third-party stakeholders (e.g., suppliers, customers, partners) are established	CP-2, PS-7, PM-2, PM-29
		ID.BE-1: The organization's role in the supply chain is identified and communicated	SR-1, SR-3

SEC is already here (Sarbanes-Oxley)

Deceptive practices that came to light in the early 2000's (Enron and WorldCom are two examples) prompted Federal Regulations to ensure transparency of documented financial and decision support (IT) activity in mainly publicly traded corporations and some privately held entities. A summary of the law's coverage follows:

- **Access:** Accessibility standards are well defined and subject to audit that cover both hard copy and electronic media. Need to access through the normal and reasonable function of departmental roles is delineated.
- **Data backup:** A backup plan and policy are absolute.
- **Security:** Breach prevention through reasonable bounds
- **Change management:** Change happens. Anticipate and plan for technologic (on-premise to cloud storage, for example), data, and software migration.

Practical Documentation Strategy

A. Adopt a Literate programming style

- Each program should read like a book
- Use indentation to de-mark sections and their statements
- Use whitespace (blank lines) to separate sections

- Start each SAS statement on a new line

B. Use Comment templates

- Documentation within a SAS program use comments, which can occur in several forms. Ideally, each Data or Proc step should have a separate documentation block as a block comment.
- **Block comments** isolate any text between /* and */ symbols. This commenting style is preferred for documentation because they may occur anywhere in a program and contain text of any composition or length.
- **Line comments** isolate any text between an * (asterisk) and the next semi-colon (;). They are useful for commenting out a single SAS statement, but less useful for documentation. They may not contain semicolons. Line comments are less useful for significant documentation.

C. Make your code self documenting

Make use of automatic system macros, for example:

- SYSFUNC(GETOPTION (SYSIN)) returns the path and name of the program
- &SASDATE returns the Date the program began
- &SYSTIME returns the Time the program began
- &SYSUSERID returns the User ID of the programmer who submitted the job

D. Build Documentation templates as paste-able text

An embedded SAS Log is a gift that can keep on giving. After a particular SAS step (PROC, DATA, etc) executes, the SAS log contains valuable context information from the session. Just cut the relevant text out and paste it into the section block comment. This provides a treasure trove of execution information for subsequent maintenance programmers. This technique can be easily applied to legacy code bases.

E. Cut/Paste Proc Dataset listings into block comments

Using proc datasets to list the variables into the LOG is another way to automatically document the contents of the PDV. Cut and paste into appropriate section documentation block.

F. Primary Document block

A block comment should be placed at the top of the program. Because it's a block comment, the format can be adapted to any useful content. A minimal template should include:

- Title;
- Author;
- Project;
- Notes such as Problem, Reasoning, etc.;
- Date created and last updated.

G. Global or Session Resource block

SAS programs often need to establish global or session resources. Gather these together in a separate block with appropriate line comment. Remember to cut and paste the generated log entries in the section documentation block.

H. Section Document blocks

Including a separate block for each major section of the program is often very useful.

- Data steps;
- Proc steps;
- block macros (%macro ... %mend).

Example 1: Primary Documentation Block

Each program should have a consistently formatted **primary documentation** block.

```
/*=====
*=====
*=====
*
* Path: C:\Users\bpeppe5\Documents\TIM\0-TI\4-CIRT\TWA\code\vti
* Module: TWA_1_UXS_A_00.sas
* Title: TMUS Web Anomaly - User X State Mapping (Part A)
* Project: TWA
* Type: SAS Program
* Author: Brett C. Peppe
* Status: 1
* Run Est:
*
*-----[ Description ] -----*
*
Read ccyyimdd_eui_all datasets created by the TWA_0_DC_03.sas(0.ti6) program and
determines a UX State map.

The EUI Splunk logs contain contain unfiltered log records
containing EUI State transactions. This program uses the values of
the following EUI files as keys:
o eui_cat
o eui_opr
o eui_msg
```

```

This program builds a Map of these triple states (UX) over any
desired time period that has EUI processed logs in the
TWA_0_DC_03.sas format.

```

```

=====
*
*
*/

```

Example 2: Globals Documentation block

It is a good idea to collect all Global operations together at the top of a program and have a separate documentation block.

```

=====
*
* [01.01] Establish Global Environment
* -----
*
* Purpose
* =====
* This procedure establishes the SAS job options, initializes global macrovars
* and cleans up the workspace.
*
* SAS Log
* =====
NOTE: Libref UXSLIB was successfully assigned as
follows: Engine: V9
Physical Name: D:\euilogs\uxstate
NOTE: Deleting WORK.PRODSAVAIL (memtype=DATA).
NOTE: PROCEDURE DATASETS used (Total process
time):
real time. 0.05 seconds
user cpu time 0.00 seconds
*
* =====
*
*/

```

Example 3: Section Documentation Block

Here is an example of a Section block comment. It contains a Purpose, Notes, a simple data map and a copy/paste of a previous run. This is ***Really USEFUL when you need subsequent maintenance.*** Using Cut/Paste to clip text from the SAS log into the comment is a quick way of documenting what the SAS engine did.

```

/*=====
*
* [0ti.01] Build EUI Index Scan Tag list (WORK.itag_0)
* -----
*
* Purpose
* =====
This procedure creates a list of primary Index Tags (ITAG) that to scan each
raw EUI records for. The intended usage is:
Given(itag_cd) Lookup(.....)
*
* Notes
* =====
Most of these appear to define master records types, so only one(1) per EUI rcd.
*
* WORK.itag_0
* =====
itag_cd itag_nr itag_cat0 itag_cat1

```

01.01	3	HTTP	GET
01.0ti	4	HTTP	POST
0ti.01	f t	FILE	FILE_NAME
03.01	1 0	JS	[AsyncRestClient.js]
03.0ti	1 1	JS	[loginApi.js]
03.03	1 t i	JS	[commonAPI.js]
03.04	1 3	JS	[greyListClient.js]
03.05	1 4	JS	[notificationService.js]
03.06	1 5	JS	[redisManager.js]
03.0ft	1 6	JS	[app.js]
03.08	1 f t	JS	[createProfileAsyncAPI.js]
03.09	1 8	JS	[SignUpAsyncService.js]
04.01	t i 1	CRIP	CUSTOMERIP=

```

* SAS Log
* =====
NOTE: The data set WORK.ITAG_0 has 13 observations and 4
variables. NOTE: DATA statement used (Total process time):

*
*=====*
*=====*
*/

```

Conclusion

By following a few reasonable documentation template practices maintenance of new and legacy SAS programs is greatly improved. Using the correct style of comments is important. Block style comments, (`/* ...*/`), are preferred because they can record any arbitrary text, provided that it does not contain an embedded `“*/”`.

We echo advocacy for an intentionally literate programing style with significant inline documentation. We conclude such styles are the best general documentation practice and highest return for knowledge worker hours expended on manual SAS documentation.

Using inline templates can assist in Audit review, or can be harvested by external tools that can review source code.

External tooling to scan SAS source, enforce formatting standards and programmatically create external documentation is in its infancy.

By following simple templates we demonstrate how productivity can be greatly improved and how this approach can be applied to the maintenance of legacy code - *the job you save may be your own*.

More Resources:

<https://support.sas.com/resources/papers/proceedings12/258-2012.pdf>

<https://support.sas.com/resources/papers/proceedings14/1617-2014.pdf>

<https://support.sas.com/resources/papers/proceedings17/1314-2017.pdf>

https://www.scsug.org/SCSUGProceedings/2005/Jackson_Maintaining%20Inherited%20SAS%20Code%20-%202019.pdf

<https://www.scsug.org/SCSUGProceedings/2003/Jackson%20-%20WritingSASRight.pdf>

Contact Information

Your comments and questions are valued and encouraged. Please include SESUG2024 in the Header line to pass the Spam filter.

Contact the primary authors at:

Brett Peppe, brett@brettsoft.com brettp@cedalionrm.com

Alan Mann, amann370@icloud.com