

## A Programming Development Environment for SAS<sup>?</sup> Programs

Tim Williams, PRA International, Charlottesville, VA  
Lee Walke, PRA International, Charlottesville, VA

### ABSTRACT

Organizing the programming effort for large projects can be a daunting task. Programming teams are under pressure to complete work accurately and rapidly, while using standard templates and macros whenever possible. Add to that several levels of documentation and a manager's need for frequent status updates and you have an administrative nightmare. A system is necessary to integrate programming documentation, validation checklists, and real-time status reports.

After deciding to centralize our project management spreadsheet, program header documentation, and validation documents, we found that we had the beginnings of a web-based programming development environment (PDE) for SAS programs. This paper describes a database-driven system for organizing SAS programs written using Active Server Pages, Visual Basic, and Javascript.

### INTRODUCTION

PRA International is one of the fastest growing Contract Research Organizations (CROs) providing programming services to pharmaceutical and biotechnology companies. Competitiveness in this industry increasingly requires globalization (Labore and Burger, 2001). Expansion of PRA to Trials Management Centers (TMCs) around the globe has fostered cooperative teamwork and the need to develop standardized, steady-state processes that apply not only to SAS program coding practices, but also to the software adjuncts used to manage the corresponding documentation and validation processes.

With an increase in workload comes the need for a strategy to increase efficiency while maintaining rigorous quality standards and adherence to the regulatory requirements of the Food and Drug Administration (FDA). A major challenge for programmers, managers, and administrators is how to efficiently organize an access all the programs and documentation required for the many projects they work on.

As drug companies race to get their products to market, this time pressure is passed along to the CROs to whom they outsource many projects. While companies attempt to shorten development cycles, high quality standards must be maintained and this requires new ways to write, document, and validate programs. How then can a CRO differentiate itself from its many competitors in the drug development arena? A successful philosophy must include the pursuit of new and innovative approaches that streamline operations and yield a competitive advantage.

The staff in our various offices all use a similar approach, but over time some individuals may add improvements to methods without enlightening the rest of the company. As Tiggemen (1997) noted, "Learning is often the cumulative result of many small improvements rather than major breakthroughs and tends

to vary depending on the amount of management attention devoted to capturing it." PRA recognized this fact in the recent creation of a SAS System Administrator position within the company. One of the roles of this individual is to synthesize improvements into a superior methodology.

### BACKGROUND

The SAS programming environment lacks a tool to organize and integrate programs with documentation and quality assurance methods. As you may well imagine, this type of activity is very specific to an industry, company, and even to the individual programmer. It is not surprising, therefore, that such a facility is not a part of the SAS system.

A solution to this predicament requires more than just software – it requires a fundamental change in how team members interface with the information. Success will only be obtained if the people who perform these tasks on a daily basis are involved in remodeling the system. Such an obvious statement may not seem worth mentioning, except for the fact that consensus building is one of the most challenging aspects of development.

The Analysis Programming Department at PRA has well-defined quality assurance methods for code development and program validation. We use a variety of tools to assist us in these activities (Figure 1), including:

- ? Detailed headers in every SAS program
- ? A "Table of Programs" (TOP) spreadsheet to organize and document programs within a project
- ? Validation Checklists in Microsoft Word<sup>?</sup>.

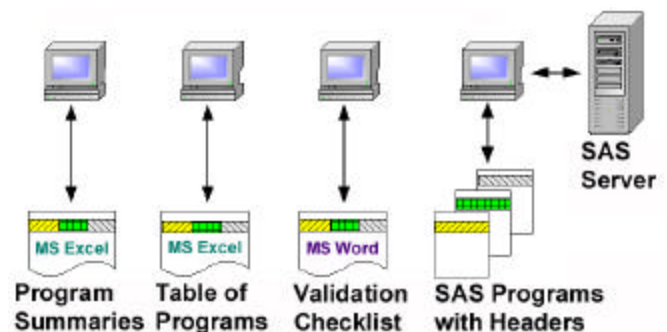


Figure 1 : SAS programs and related files prior to integration.

## Program Header Documentation

Documentation of programs is one of the least favorite activities of programmers and is often put off until the last minute. As a result, documentation may be sparse, incomplete, inaccurate, or nonexistent. By making this task an integral part of the project setup and program validation process it can be a more seamless and less burdensome activity.

Gill (1997) states that narrative documentation of program code is essentially *internal* (such as program headers and in-line comments) and *external* (such as our Table of Programs spreadsheet). Program headers serve not only as a guide to our programmers, but also to our clients who may request the source code as part of the deliverable product. Hence, any documentation stored externally to the program code must become part of the program or be adequately referenced and accompany it.

PRA's Standard Operating Procedures define a structure for our program headers (**Appendix 1**) that includes:

- ? Client and study name
- ? Author and code completion date
- ? Program abstract
- ? Author and date of code validation
- ? List of input and output files
- ? List of macros called
- ? Notes
- ? Amendments (and amendment validation).

## The Table of Programs (TOP)

Our standard TOP document (**Appendix 2**) allows programmers to rapidly determine program status and dependencies, as well as providing an overview of the entire project. Information for each program must be entered, including most of the items already mentioned in the program header. A summary page allows the team leader to identify project status by summarizing how many programs have been written and validated.

## Program Validation Checklists

These checklists are Microsoft Word documents that vary depending on the type of SAS program. Clearly, a program that involves complex data derivations must undergo a different validation procedure than one that simply lists data points. In addition to the checklist items, certain redundant elements must be entered into these documents, such as: program author and date, validation author and date, amendment author, etc.

No off-the-shelf solution was available that would allow the integration of our header documentation, TOP, and validation documents. The choice to develop our own customized solution was an easy one to make and the concept of PRA SAS Manager (PRASM) was born.

## **COMPLIANCE WITH INDUSTRY REGULATIONS**

This paper does not deal with the validation of the SAS system, but instead focuses on how to make programming with SAS

more compliant with industry regulations, specifically those in the pharmaceutical industry.

## 21 CFR Part 11

The FDA provides compliance guidelines for the pharmaceutical industry. "Guidance for Industry. Computerized Systems used in Clinical Trials," (FDA, 1999) addresses the requirements of "Electronic Records/Electronic Signatures" rule of 21 CFR Part 11 (FDA, 1997) and applies to any source documents created; be they hard copy entered into a computer system, direct human entry into the system, or generated automatically by a computerized system. Therefore, SAS programs are governed by these regulations when producing data for submission to the FDA.

Maintaining an audit trail is a critical requirement. The FDA (1999) defines an audit trail as:

"...a secure, computer generated, time-stamped electronic record that allows reconstruction of the course of events relating to the creation, modification, and deletion of an electronic record."

"Changes to data that are stored on electronic media will always require an audit trail, in accordance with 21 CFR 11.10(e). Documentation should include who made the changes, when, and why they were made."

These requirements highlight why a compliant system must: (1) allow access only by authorized personnel and (2) provide a detailed log of their activities that result in changes to data or documentation.

## System Security

"Security measures should be in place to prevent unauthorized access to the data and to the computerized system." (21 CFR Part 11, 11.10(k) )

By extension, only authorized team members should access a project's programs and documentation. PRASM will capture the Windows 2000 username from the initial workstation logon. The logon name will be looked up in a security table (maintained by the PRASM Administrator) and access to projects is granted to approved staff members.

The staff altering electronic records "...should not be able to modify the audit trails." (FDA, 1999) PRASM will make this possible by storing all actions in a secure log file for audit purposes.

## Activity Log

Capturing an audit log is of utmost importance in complying with 21 CFR part 11:

"Use of secure, computer-generated, time-stamped audit trails to independently record the date and time of operator entries and actions that create, modify, or delete electronic records. Record changes shall not obscure previously recorded information." (FDA 21 CFR Part 11, 11.10 (e) )

A SAS programming team must accurately record actions that result in changes to data. Many SAS programming environments strive toward this goal, but it often remains out of reach. Documentation of changes to programs is left in the hands of individual programmers, whose compliance may fall short of the required standard. A more automated and accurate audit trail is a desirable goal when moving toward FDA compliance.

As stated by Sporon-Fiedler et al. (2002), “A computer system like SAS is a mixture of development and data processing environment with the capability of producing self-contained systems.” They further state, “Proving that a log and output file has its origin from an exact copy of a specific program requires a finely synchronized monitoring tool to prove that no tampering has taken place... Proving that a database (e.g. SAS datasets) is in complete control, requires that all modifications (updates) are tracked.”

The FDA guidance details how the date and time of operator entries must be separately recorded for all actions that create, modify, or delete electronic records. While this has become an integral part of the data entry process in many organizations, compliance is often less than optimal for SAS programmers when they make changes to programs that alter data sets or summary displays (tables, listings, and graphs). Such information must be recorded when alterations are made to programs that result in changes to output.

We must ensure that all program amendments are documented and that no alterations remain unaccounted for. PRASM will do this by comparing the file system date on the SAS program with the date of the last entry in the database for the corresponding program. If the file system date is more than one day younger than the date in PRASM, then warnings are issued alerting the lead programmer to investigate the offending program.

Dating of all changes will be handled automatically by the system, which relies on a validated system clock. PRASM complies with the FDA (1999) requirement that, “Dates and times are to be local to the activity being documented and should include the year, month, day, hour, and minute.” All actions by staff using the system are logged with staff identifiers and detailed time stamps.

When programmers work across TMC’s in multiple time zones, the time stamps will reflect local time where the change was made. As a browser-based system, PRASM will use a Javascript clock for time stamping. The clock derives the time from the clock on the programmer’s workstation, which in turn is synchronized with the validated time-server on their local network. Hence, all times in the system are local to the programmer making the change.

### Version Control

A compliant system contains:

“Revision and change control procedures to maintain an audit trail that documents time sequenced development and modification of systems documentation.” (21 CFR Part 11, 11.10 (k) )

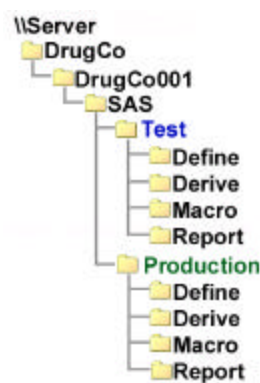
SAS programming lacks a strict means of version control (Williams, 2002) because it takes place outside of a formal Integrated Development Environment (IDE). Implementation of such a controlled environment adds additional overhead to the process and may not be readily accepted by some programmers. Making version control an integral part of program documentation and validation procedures will avoid these perceptions. Version control becomes part of the programming process with little or no additional input from individual programmers.

PRASM will not include a full-scale version control solution. It will, however, provide automatic backup of files. The time-stamped backups can be compared using version comparison tools, such as the freeware utility ExamDiff (PrestoSoft, 2002), in conjunction with a log of program amendments that exist as part of the program header.

### Separating Development and Production Environments

Another important aspect of compliance with FDA guidelines is the separation of development or *testing* code from finalized *production* code. By using separate folder structures for finalized, validated programs (**Figure 2**), we can ensure that the programs and the output they produce are synchronized. Once programs have been validated the system will allow the programmer to *move the file to production* at the click of a button. If a file requires editing, it is moved down from production and, only after the edits have been validated, will be allowed to move back into production. Programs, log files, and output will be automatically archived into a time-stamped shipment folder each time output is sent to a client.

The key points of compliance with 21 CFR Part 11 – data security, audit trails, and management of programs and documentation, can all be attained to some degree through the use of PRASM along with a standardized management approach. Much of PRASM’s approach to the data security issue is indirect, controlling access to the programs that produce SAS datasets,



**Figure 2. Production and Development (Test) directories for hypothetical project 'DRUGCO001.'**

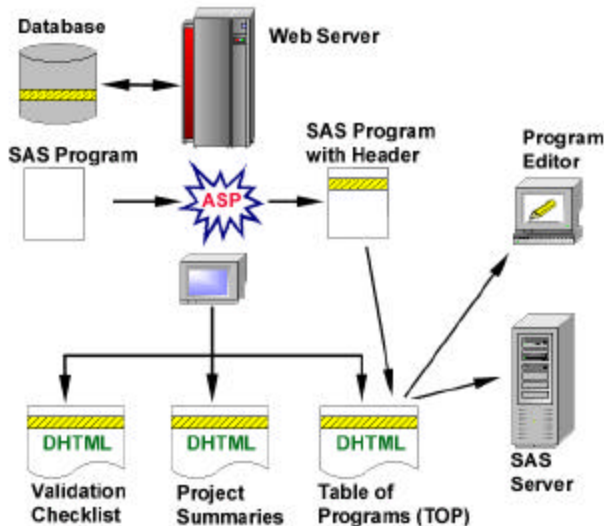
instead of controlling access to the SAS datasets themselves. Audit trails are dependent on programmers accurately documenting amendments to programs. Automatic backups of programs provide a means of comparing previous versions of code. Documentation exists by integrating the program header

information with validation documentation in a single location - the PRASM database.

## DEVELOPING THE SYSTEM

### CONCEPTUAL FRAMEWORK

Related elements of the coding process will be integrated into a web-based system as illustrated in **Figure 3**. A centralized database will hold information about each program for display and updating through a dynamically generated web interface. These pages will also provide the ability to launch programs into a code editor or submit them to various remote SAS sessions.



**Figure 3 :** Integrated approach to managing SAS programs and related files.

Our goal is not to produce a full-scale Program Development Environment (PDE). We already have tools for code editing and syntax highlighting (UltraEdit (Mead, 2002) and the SAS Enhanced Editor). Other software adjuncts are available for diagramming program flow (ComplementSoft, 2002) and our Standard Operating Procedures govern our code writing methodology. The main goals of our new system are to:

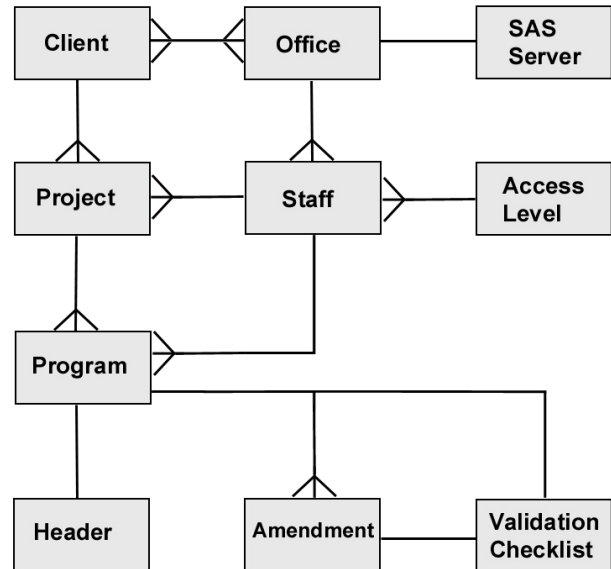
- ? Standardize and reduce redundancy in validation and documentation procedures
- ? Link programs to their documentation and validation information
- ? Facilitate rapid programming using inter-office programming teams
- ? Rapidly communicate project and program status
- ? Increase compliance with FDA requirements.

#### Database Schema

Our system is based on a hierarchical database structure (**Figure 4**). The diagram illustrates that a Client may have one or more Projects assigned to offices at PRA. Each Project contains several Programs that have unique Header information. In addition, each program may have one or more Amendments and at least one Validation Checklist (more than one if there are amendments). A Client may have Projects at more than one Office. Staff at each office are assigned an Access Level

within the PRASM system and are associated with Projects and Programs as they contribute to coding and validation activities.

The representation in **Figure 4** is somewhat simplified and will continue to evolve with the application. Since a Project is associated with an Office, SAS programs can be automatically submitted to the local SAS Server at that particular office, thereby helping to reduce traffic on the company's network.



**Figure 4 .** Simplified PRASM database schema.

The scope of the project quickly expanded from merely a documentation tool to a more integral part of our programming methodology and philosophy. It will allow us to review our processes and answer questions such as: "How many hours are required for a Phase III Oncology trial versus a Phase III Anti-Infectives trial?"; "Which offices are under or over staffed?"; "Are the code writing and validation tasks shared equally among programmers?"

### PUTTING THE PIECES TOGETHER

We are following, "7 Practices for Excellent Software," (Barnes Nelson and Grasse, 2002) during construction of the application. Early development has started on a workstation running Personal Web Server and a Microsoft Access<sup>7</sup> database. We are applying the principle of "Orthogonality" by using structures and naming conventions that will facilitate later porting to an Oracle database and ASP.NET.

Our initial focal point was one of the most critical areas of the system: the linking of program header documentation to the SAS program. Each time the program header page of PRASM (**Appendix 3**) is updated, the ASP code reads in the appropriate SAS program file and starts searching for a standard marker line:

```
/*!--DO NOT EDIT THIS LINE OR ABOVE--
!!*/
```

When it finds the marker, all code above that line is discarded (deleting the now outdated header information) and the new

header information is written in its place. The application appends the remainder of the program below the marker line. Only the header is updated and the program code itself remains intact. A time-stamped backup copy of the program is also created and archived.

Once the program header issue was solved, we turned back to the topic that had started our development in the first place, the linking of validation checklists to each SAS program. Our Microsoft Word checklists account for additional redundancy, so we can further streamline our processes by making them part of the new application (**Appendix 4**).

Programmers and team leaders also need a quick way to check the status of a program and determine if it:

- ? Has been written
- ? Is ready for validation
- ? Has been validated
- ? Is *in production*.

This will be available to programmers at a glance on a web page that provides status *traffic lighting* for each program in a project. (**Appendix 5**).

Once the documentation is linked to the program and its status is easily determined, we can launch a selected program into a code editor and remote SAS sessions. This presented a technical obstacle, since browsers are generally not permitted to launch desktop applications due to security concerns. We overcame the challenge by using "Launch-In-IE" (RockinFewl, 2002). After installing the supplied .DLL and making minor changes to the Windows 2000 registry, we set up a trusted URL for the PRASM website. We can now send SAS programs directly into our code editor and to local or remote SAS sessions from a web browser. Since our database associates projects with offices, PRASM will automatically determine to which server it should remote submit programs.

## STATUS

Our staffing and workload constraints lead us to adopt the staged delivery model of software development (McConnell, 1996). Staged delivery, also known as incremental implementation, makes content available to users in successive stages as the application is developed. In this way our staff will see tangible benefits of the system without having to wait for its full implementation.

In the current version of PRASM, programmers are presented with an initial screen (**Appendix 6**) where they select a client and then drill down to an individual project. From there, a list of all programs within a project and their individual status appears on a web page that resembles our original Table of Programs (**Appendix 5**). On this page programmers can quickly determine the status of a program based on a series of traffic lighting indicators. They will know at a glance if a program has been completed, if it has been validated, and if it has been moved from the development folder to production. At the click of a button they can: edit and update the program header (**Appendix 3**); fill out the program validation documentation (**Appendix 4**); move the program into a production directory; send the program into the program editor; submit it to a remote or

local SAS session; or archive all programs, logs, and outputs into an archive directory.

At this time we have identified a small project on which to pilot test the system. A small group of programmers will run through the project, provide feedback, and assist in further refinements to the application.

## FUTURE DIRECTIONS

Future enhancements to PRASM may include the following features:

- ? Parse SAS program log files to identify ERROR, WARNING and user-defined problem statements
- ? Add reporting features for time periods: number of programs written, checked, validated; number of programs that required amendments or had problems; number of shipments sent to clients; office and staff workloads; etc.
- ? Forecast the amount of hours required to finish projects
- ? Customizable templates for program headers and validation checklists
- ? Track activities to help design in-house training and manage resources.

Since PRASM will already read-in and parse entire SAS programs, it is possible to obtain additional value from in-line comments. This could follow a model similar to the Javadoc approach (Sun Microsystems, 2002) where in-line comments are used to produce HTML-based documentation. Javadoc-style inline comments are a natural choice, since their format mirrors those of the *delimited* comment available in SAS:

```
/**
 *
 * This is the description part of a doc comment
 *
 * @tag Comment for the tag
 *
 */
```

The benefits of implementing such a system are debatable and are not slated for incorporation into PRASM at this time. The possibility of disconnects between delimited in-line comments and the code they relate to may cause more confusion than potential benefit. However, the concept remains intriguing.

## CONCLUSION

The PRA SAS Manager System is currently in its infancy but shows great promise. What started out as an idea to link program and validation documentation has grown into a much larger endeavor.

Our program documentation will soon no longer exist as disconnected elements, but will instead share fields with the Table of Programs and SAS program headers in a database. Redundancy of data entry will be greatly reduced as the entire process becomes streamlined due to the centralization of

information. Programmers can spend more hours on client-oriented project work and less time laboriously filling out redundant information.

## REFERENCES

Barnes Nelson, Greg and Grasse, Danny (2002), *(Web) Software Development: Best Practices for Developing Enterprise Applications*. Proceedings of the 27th Annual SAS Users Group International Conference, Orlando, FL.

ComplementSoft (2002), *ASAP. Productivity Tool for SAS Users*.

<http://www.complementsoft.com/product.htm>

Food and Drug Administration (1999), *Guidance for Industry, Computerized Systems used in Clinical Trials*.

Food and Drug Administration (1997), *21 CFR Part 11, Electronic Records; Electronic Signatures; Final Rule*. Federal Register Vol. 62, No. 54, 13429.

Gill, Paul (1997), *The Next Step. Integrating the Software Life Cycle with SAS Programming*, Cary, NC: SAS Institute Inc., 384 pp.

Labore, John M., Rogers, Melinda S. and Steven D. Randolph. (2001), *Singing Cowboys, Fast Horses, and Team Roping: Keeping SAS Users Calm and on the Trail*. Proceedings of the 26<sup>th</sup> Annual SAS Users Group International Conference, Long Beach, CA.

McConnell, S (1996), *Rapid Development: Taming Wild Software Schedules*. Microsoft Press. 647 p.

Mead, Ian (2002), *UltraEdit Text Editor*.

<http://www.ultraedit.com/>

PrestoSoft (2002), *ExamDiff - Visual File Comparison Tool*.

<http://www.prestosoft.com/examdiff/examdiff.htm>

RockinFewl (2002), *Launch-in-IE. Web pages can start applications. Securely*. <http://whirlywiryweb.com>

Sporon-Fiedler, Gustav, Lassen, Marie and Lundbewck, H. (2002), *SAS Coexistence with FDA 21 CFR Part 11, How Far Can We Get?*. Proceedings of the 2002 Annual Conference of the Pharmaceutical Industry SAS Users Group (PHARMADSUG, Salt Lake City, UT.

Sun Microsystems (2002), *The Javadoc Tool Homepage*. <http://java.sun.com/j2se/javadoc/>

Tiggeman, Rolf E. and Sabel, Hermann (1997), *An innovative Concept in Pharmaceutical Drug Development*. Drug Information Journal, Vol. 31, pp. 119-124.

Williams, Tim (2002), *A Version Control Kluge for SAS Programs - Using SAS*. Proceedings of the 27<sup>th</sup> Annual SAS Users Group International Conference, Orlando, FL.

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 registered trademarks or trademarks of their respective companies.

## **ACKNOWLEDGMENTS**

We wish to thank the Analysis Programming staff at PRA International for their continuing feedback and assistance.

## **CONTACT INFORMATION**

Your comments and questions are valued and encouraged. The authors may be contacted at:

Tim Williams  
SAS System Administrator  
Email: [WilliamsTim@PRAIntl.com](mailto:WilliamsTim@PRAIntl.com)

Lee Walke  
Director of Analysis Programming  
Email: [WalkeLee@PRAIntl.com](mailto:WalkeLee@PRAIntl.com)

PRA International  
4105 Lewis and Clarke Drive  
Charlottesville, VA 22902  
Telephone: 434.951.3000  
Fax : 434.951.3001  
Web: [www.praintl.com](http://www.praintl.com)

## APPENDIX 1 : EXAMPLE SAS PROGRAM HEADER

```

/*****
STUDY          : Drug Co, DRUGCO001
PROGRAM NAME   : m_demog.sas
PURPOSE        : Derive Demographic data set.
PROGRAMMER     : Tim Williams
DATE           : 08/12/2002
QC             : BY,DT : Jimbo Jones, 08/13/2002
INPUT FILES    : //FileServer/DrugCo/DrugCo001/data/datasets/raw/demog.sd2
OUTPUT FILES   : //FileServer/DrugCo/DrugCo001/data/datasets/final/demog.sd2
MACROS USED    : %CI, %AGECALC
NOTES          : Derived data set is used in l_demog.sas and t_demog.sas
                :
AMEND[1]BY,DT  : Nelson Muntz, 08/20/2002
AMEND[1]DETAILS : Added age calculation to Demog data set
AMEND[1]QC BY,DT: Jimbo Jones, 08/22/2002
                :
Header last updated 11:19 Thursday, August 08, 2002
Copyright (C) 2002, Pharmaceutical Research Associates, Inc.
All Rights Reserved.
*****/
/*****
/!!--DO NOT EDIT THIS LINE OR ABOVE--!!*/

```

## APPENDIX 2 : EXAMPLE TABLE OF PROGRAMS (TOP)

PROGRAMMING STATUS: TABLES													
DrugCo DrugCo001													
Program Name	Title/Description	Requirements		Output	Programmer		Program Validation					Comments	
		data sets	macros/ progs		Initials	Date Completed	QC Level*	Assigned To	Date Completed	Amendment	Amendment QC		
t_demog.sas	Table 1A Demographic and Baseline Characteristics	librarg.demog	%PageNum	t_demog.txt	TV	12-Aug-02	4	JJ	13-Aug-02	NM, 18-Aug-02	JJ, 20-Aug-02		
t_enroll.sas	Table 2 Patient Enrollment	librarg.demog	%PageNum	t_enroll.txt	TV	14-Aug-02	4	NM	16-Aug-02	None			
t_disp.sas	Table 3A Patient Disposition	librarg.disp	%PageNum	t_disp.txt	JJ	12-Aug-02	4	TV	16-Aug-02	None			
							QC Levels						
							1	Simple Formatting			Second Programmer not required		
							2	Simple Formatting			Second Programmer required		
							3	Complex Data			Second Programmer not required		
							4	Complex Data			Second Programmer required		
							5	Very Complex data Manipulation			Two programmers independently produce identical results		

**APPENDIX 3 : PROGRAM HEADER (PRASM WEB PAGE)**

The screenshot shows a web browser window titled "PRA SAS Manager - Microsoft Internet Explorer provided by PRA International". The address bar shows "http://williamstim2k/prasm/PRASASManager.asp". The page header includes the "p/a SAS Manager" logo. On the left sidebar, user information is displayed: "User: Tim", "Access Level: Administrator", and "Today is: 6/13/2002 6:52:16 PM".

The main content area is titled "Program Header for Drug Co, DRUGCO001 : autoexec.sas". It contains a table with the following details:

Program Name	autoexec.sas
Title/Descr.	Define libraries, options and global macro variables
Last modified	6/13/2002, 5:51:55 PM
Header Last Updated	6/14/2002 (Date difference: -1)
Status	Ready for QC
Program Type	Define
QC Level	Simple. 2nd Programmer required
Root Path	\\Server\DrugCo\DrugCo001\SAS\Define\
Sub Dir.(no "\")	define
Full Path	\\Server\DrugCo\DrugCo001\SAS\Define\autoexec.sas
Purpose	Define all libnames, options and global

**APPENDIX 4 : VALIDATION CHECKLIST (PRASM WEB PAGE)**

The screenshot shows a web browser window titled "PRA SAS Manager - Microsoft Internet Explorer provided by PRA International". The address bar shows "http://williamstim2k/prasm/PRASASManager.asp". The page header includes the "p/a SAS Manager" logo. On the left sidebar, user information is displayed: "User: Tim", "Access Level: Administrator", and "Today is: 6/13/2002 6:27:41 PM".

The main content area is titled "Validation Checklist". It includes a printer icon and the text "21x24x36 version". Below this, the client and project information is shown: "Client Drug Co Project DRUGCO001, Program autoexec.sas".

The section is titled "Program Code" and contains a table with the following validation items:

Item	Yes No N/A			Changes Needed?		Remark
	Yes	No	N/A	Yes	No	
Header is complete and accounts for amendments (if any?)	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	
Body comment present before each code section?	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	Acceptable, but additional comments would code review in
All path names, libnames, global macro variables in autoexec.sas?	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	
Formats in the format sas or similar (none defined in	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	

**APPENDIX 5 : TABLE OF PROGRAMS (PRASM WEB PAGE)**

The screenshot shows the PRA SAS Manager web interface in Microsoft Internet Explorer. The browser address bar shows the URL: http://williamstim2kj/prasm/PRASASManager.asp. The page title is 'PRA SAS Manager'. The user is identified as 'Tim' with 'Administrator' access level. The date and time are 'Today is: 6/11/2002 10:44:26 PM'. The page content is for 'Drug Co Project DRUGCO001' and displays a table of programs.

Program Type	Program Name	Title/Description	Status <small>Color Code</small>	QC <small>Color Code</small>	Production? <small>Color Code</small>
Define	autoexec.sas <a href="#">edit</a> <a href="#">Submit</a>	Define libraries, options and global macro variables	<span style="color: orange;">●</span>	<span style="color: red;">●</span>	<span style="color: red;">●</span>
Derive	m_ae.sas <a href="#">edit</a> <a href="#">Submit</a>	Derive Adverse Event data set	<span style="color: green;">●</span>	<span style="color: green;">●</span>	<span style="color: green;">●</span>
	m_demog.sas <a href="#">edit</a> <a href="#">Submit</a>	Derive Demographics data set	<span style="color: green;">●</span>	<span style="color: green;">●</span>	<a href="#">Move</a>
Listing	l_ae.sas <a href="#">edit</a> <a href="#">Submit</a>	Listing of Adverse Events	<span style="color: blue;">●</span>	<span style="color: grey;">●</span>	
	l_demog.sas <a href="#">edit</a> <a href="#">Submit</a>	Listing of Demographic Data	<span style="color: grey;">●</span>	<span style="color: grey;">●</span>	

**APPENDIX 6 : PRASM HOME PAGE**

The screenshot shows the PRA SAS Manager home page in Microsoft Internet Explorer. The browser address bar shows the URL: http://williamstim2kj/prasm/PRASASManager.asp. The page title is 'PRA SAS Manager'. The user is identified as 'Tim' with 'Administrator' access level. The date and time are 'Today is: 6/13/2002 7:58:10 PM'. The page content is organized into three main sections: 'Manage Projects', 'Administration', and 'Reports'.

**Manage Projects**

- Drug Co
- 

**Administration**

- 
- 

**Reports**

- 
-