

Using SAS[®] Drug Development as a Report Management Application

Barry R. Cohen, Planning Data Systems, Inc.

ABSTRACT

Many statisticians and statistical programmers in the pharmaceutical industry will first come to know SAS Drug Development as a product that addresses their regulatory-compliance issues (auditing, versioning, and security) as they develop their on-going analysis programs, data, and documents for NDA filings. However, the product provides a full, flexible processing environment that can be used in other ways. In this paper, I examine standard features of SAS Drug Development that allow it to serve as a Web-enabled report management application for a library of SAS-based report programs. The application covers typical functions such as (1) loading SAS macro report programs to the environment; (2) building report parameter-solicitation screens; (3) providing a user interface to select report programs, set report parameter values, and execute reports; (4) managing report output sets and providing a user interface for output file viewing. Emphasis is on what SAS Drug Development can provide for a report management application “out of the box” without a user-customization of the application through the product’s API.

INTRODUCTION

SAS DRUG DEVELOPMENT

Statisticians and statistical programmer in clinical bio-statistics departments of pharmaceutical companies develop electronic versions of analysis and report programs, data, and documents on an on-going basis in support of New Drug Application (NDA) filings with the FDA and other agencies globally. I will refer to this as the “bio-statistics on-going work process” in this paper.

The regulatory agency (FDA) requirements for this computer-based work process have changed substantially in the last few years. The FDA, through the regulations known to the industry as “21CFR11”, now has specific requirements for building and validating the software systems that are used to support the bio-statistics on-going work process. And they have specific requirements for how the various program, data, and document files produced are to be handled in terms of versioning, auditing, and controlling access to them. This paper will not cover the details of these requirements. The user is referred to the References section at the end of this paper for citations for two “White Papers” by the SAS Institute that discuss this subject at length in the context of describing the SAS Drug Development product.

In general, SAS Drug Development (SDD) is a product that provides a 21CFR11-compliant platform for conducting the bio-statistics on-going work process. The idea is that if the statisticians and statistical programmers of a pharmaceutical company do their work on the SDD platform, they will be meeting the 21CFR11 requirements as they work. This is because SDD will version, audit, and secure access to their program, data, and document files and folders, in the background, as these files and folders are created and changed. In essence, when using SDD, the statisticians and statistical programmers do the same work they have always done, and they do it in the same way they have always done it. As a primary example, they have always had a repository for the electronic files and folders they have created. Formerly, this repository has typically been comprised of a directory or folder structure of some sort on a server system of some sort (e.g. NT servers, UNIX servers). But now the SDD platform provides the repository for the electronic files and folders, and SDD handles these files in the 21CFR11-compliant way in which they need to be handled today.

There is much more to say in a full description of the SDD product; however, this brief, top-level, description covers the primary aspect of the product. And this aspect will be the first focus of most statisticians and statistical programmers who use SAS Software in their clinical bio-statistics work.

SDD AND REPORT APPLICATIONS

The bio-statistics on-going work process in many pharmaceutical companies includes the development of report programs, written with SAS software, which are executed to produce tables that directly or indirectly become part of a Clinical Study Report submitted to the FDA as part of an NDA. I refer here primarily to count and listing type tables, which tend to be (but are not limited to) Safety Tables, as opposed to the statistical tables that support the efficacy analysis. Many of these report programs are standard, meaning they are applicable to most of the drug projects and NDA filings with which the company is involved. And the individual projects often also have a set of report programs that are specific to the given project. These report programs typically produce multiple output files, such as Log, List, data set, RTF or PDF rendering, plain text, etc. And they need to be run repeatedly during a project, which leads to generation of many instances of their output file sets. And,

although these report programs are written by SAS programmers, they are often executed by non-programmer users on the project teams. Thus, as part of the bio-statistics on-going work process, there is often a need for a report application, or report management application if you will, to handle this reporting effort.

I had an opportunity recently to consider how SDD might function as just such a report application. I thought that other pharmaceutical SAS users might be interested if I shared some thoughts about this recent project experience. One reason for this interest might be because this view of SDD is not the first way that most bio-statisticians and bio-statistical programmers who use SAS Software in their clinical bio-statistics work will think about SDD.

My project involved consideration of version 2.0 of SDD; however, the production version of SDD v2.0 had not quite released during my project. Instead, I had limited access to an early release of v2.0 which had most but not all the features planned for SDD v2.0. But I did not have the fuller look at the production release that was anticipated when this paper was planned. As a result, I consider my thoughts herein to be preliminary. I expect to have better access to the production version of SDD v2.0 after its release, and thus expect to have additional material in the presentation of the paper at the PharmaSUG conference.

Two major sections follow in this paper. In the first one I describe the general requirements for a report application of the type to which I refer herein. And in the second one I describe how this report application could be built and operate on the SDD platform. My focus is on how the standard, out-of-the-box features of SDD could be used to provide the report application. It is also possible for a user who knows Java or Javascript to extend SDD functionality through its API, but this is not my focus today.

REPORT APPLICATION REQUIREMENTS

Following are general requirements for a report application of the type introduced above. Such an application could be found in many clinical bio-statistics or related departments at pharmaceutical companies today.

END-USER REQUIREMENTS

The report application must support the execution of report programs written with SAS Software. These reports are typically parameter driven. SAS programmers write them but they are run by non-programmer end-users. This means that a user interface (UI) is required to allow the end-users to see the available reports, (both standard cross-project reports and project-specific reports), select certain reports, set the parameters for the selected reports, submit the reports for execution, and view the report outputs. Finally, end-users need to be able to save the work they did in a work session and retrieve it for continued work in a subsequent work session.

SAS PROGRAMMER-USER REQUIREMENTS

SAS programmer-user activities in the environment are different than those of end-users. Programmers must be able to load their report programs (developed elsewhere) to the environment. They must also be able to load the definition of a unique parameter-setting screen for each report program, and insure that there is a link between the input fields on the parameter-setting screen and the macro variables in the SAS report program. And programmer-users must be able to execute their programs in the environment in Test mode before they are made available to end-users in Production mode.

ADMINISTRATOR-USER REQUIREMENTS

The application will be used by many end-users and programmer-users who will work on many different projects. Thus, there will be need for an administrative function in the environment. The application must allow the admin-user to create a set of project folders for each new project being added to the environment. These folders will hold programs, input data, and report output file sets. The application must require userids and passwords for access to the overall environment, and allow the admin-user to create these. The application must also allow the admin-user to grant individual userids and user groups access to, and certain permissions in, various areas within the environment, including especially the individual drug project areas. And these accesses and permissions must be grantable through the use of roles to which userids and user groups are assigned.

OTHER REQUIREMENTS

A few other requirements are applicable to all of the above user-types, and are presented in this section.

Architecture: In today's application landscape, the report application is likely to be used by many users, possibly at several sites within the organization. It is thus generally preferable today to run a thin client and execute the report programs server-side. And this generally involves a web-enabled application with a browser-based interface on the client-side.

Regulatory: The report application must meet the 21CFR11 requirements for versioning, auditing, and securing access to the files and folders that it handles. And the company must be able to demonstrate that the report application itself was developed in a validated manner, where the standards for validated system development are also part of the 21CFR11 requirement.

Flexible input data source: Many pharmaceutical companies maintain their clinical data in a database management system and they input database tables directly to their SAS report programs. Others have the data in SAS data sets at the time of input to the SAS report programs. The report application must allow for input from either data source.

HOW SDD MEETS THE REQUIREMENTS

SDD is both: (1) a software application that already provides some of the pieces you need for your report application; (2) a platform which you can configure to provide some application pieces yourself. This is fundamental to understanding how SDD can be used to provide a report application as described above. You will see this throughout this description of how SDD can meet the report application requirements.

ARCHITECTURE

SDD is a web-enabled server-side application. Users interact with the server-based SDD application through a client-side internet browser window. By using SDD as the platform for your report application, you are adopting the n-tier, thin-client, web-enabled architecture that is contemporary today. (And this is true both for use of existing features of the SDD product out-of-the-box, and for features you might add to the platform through the API).

USER INTERFACE

User folder navigation: The overall UI for folder navigation within SDD v2.0 is Explorer-like. This means you will see a folder tree structure on the left side of the screen for general navigation through the user folders of the application, and folder contents will be visible on the right side. And users are free to construct virtually any folder structure they want and place whatever files they want in these folders. When you use SDD as the platform for your report application, you are adopting this Explorer-like UI for user folder navigation.

Custom screens for native SDD features: SDD also has a series of pre-programmed features which are selectable via a menu bar across the top of its basic graphical interface. An example would be the native facility for userid management. Each of these native features has a built-in interface comprised of one or more screens. These built-in interfaces have been programmed in Java and have a look and feel common for browser-based screen content today.

Thus, the overall UI for a report application built on the SDD platform will be part Explorer-like (for user folder navigation), and part a look and feel that is common for browser-based screen content today, (for the pre-built SDD features that you engage for your application).

REGULATORY COMPLIANCE

The files and folders you create on or load to the SDD platform become SDD objects. SDD then provides built-in versioning and auditing for its objects. This is one key way that the 21CFR11 regulatory compliance is built into the platform. So, if you build your report application on the SDD platform, you can create, modify, and delete whatever files and folders you need as you do your reporting work, and SDD will version and audit all these objects in the background as you work. For example, the following types of items that you will work with in your report application will be objects to SDD:

- Folders
- SAS program files
- SAS data set files
- External data files (e.g., spreadsheet files)
- Report program outputs such as Logs, Lists, SAS data sets, RTF and PDF files, graphs.

SDD also has built-in capability to control access to the overall application on the server-side platform where it resides, and to further control access to particular locations within the platform such as your individual project folders. This is another key way that SDD contributes to the 21CFR11 regulatory compliance needs of your report application.

Finally, the code behind the SDD product has itself been validated by SAS to meet FDA requirements. Thus, when you use this SAS product as the basis of your report application, you are assured of using a validated product. The only part of your report application you will need to validate will be the individual SAS macro report programs that your programmer-user staff write and add to the environment.

CONFIGURING THE FOLDER STRUCTURE ON THE PLATFORM

Presumably, your report application will have a folder structure to support the many drug projects that will use the application. And you will design this structure to allow the addition of new projects over time to an existing folder hierarchy that is meaningful to your department. For each new project added, a set of folders will be created. And these folders will hold

project-specific SAS report programs, input data sets (if not using a separate clinical database such as in Oracle), and report program output files.

SDD provides a folder object on the platform and you can create any number of folders where needed and in whatever hierarchy is needed. The system administrator (admin-user) will have the ability to create these folders anywhere. The admin-user can also grant authority to certain other users to create these folders in certain areas. For example, the admin can create a basic project folder structure for a new project, and then grant the project team members the authority to create additional folders in their project area. Further, these folders can be created manually, such as via the "New Folder" operation in Explorer or via the folder copy and rename operations. Or, you can develop a custom (SAS) program that can generate the folders when it executes on the SDD platform.

SDD also has the ability to function as a regulatory-compliant platform for your department's entire set of processing, not just the report processing being considered now. In fact, this is the main focus of SDD. Thus, it is possible that the folders you create for your report application will be a subset of a larger folder set created on the platform for all of the department's processing. It might be helpful to keep this in mind as you read further.

Summary points:

- SDD does not come with a folder structure for a report application. Rather, SDD provides tools for you to create the folder structure you want manually, and it will allow you to create the folder structure programmatically (with your own SAS program you develop and then execute on the platform).
- SDD provides admin tools to define userids and passwords, user groups, and roles, and to grant users access to and privileges in certain folders.
- SDD provides authentication of userid access and privileges throughout the platform for: overall entry to the platform, individual folder entry, and individual file entry.
- Your report application folders may be a subset of, and integrate into, a larger set of folders on the SDD platform that represent a regulatory-compliant repository for your department's entire set of processing.

LOADING PROGRAM FILES AND OTHER FILES TO THE PLATFORM

SDD does provide a library of its own pre-programmed drug clinical reports; however, I expect that most organizations using SDD for a report application will already have a library of custom SAS report programs to be loaded up-front to the report application on the SDD platform. I also expect that most organizations will continue to develop their custom SAS report programs outside of the SDD platform and then load the program files to the platform on an on-going basis. This is because the program editor provided with SDD v2.0 is not as robust as other program development environment, (such as native SAS with the SAS Display Manager), that are available to an organization. SAS has plans to provide enhanced program editor capabilities in SDD post v2.0 and this will be helpful for some organizations. But I expect that those organizations which have already assembled sophisticated program development environments will continue to develop their report programs in their external (to SDD) environment.

In essence, I expect that most organizations will use SDD as an application to manage a reporting process that uses custom - built reports which are built outside of the SDD environment and then loaded to the platform. In this scenario, the programmer-users will need the ability to load their SAS reports to the platform and register them for execution on the platform by end-users. SDD provides full capability to load program files (and other files) to the platform. SAS programs are treated as code segment objects on the SDD platform. When you load your externally based SAS program files to the platform, you define them as code segment objects.

SDD also allows you to load existing SAS data sets to the platform, as data table objects, to be used as input to your SAS report programs. And if your organization instead uses database tables as a direct input data source to your report programs, SDD v2.0 will provide the capability to do this through its remote generic access engine that can connect to Clintrial, Oracle, Oracle Clinical, and a separate SDD instance. Note, though, that the various remote access engines do not all ship with SDD v2.0. My understanding is that some are still in development and will be available later.

Summary points:

- You will likely continue to develop your SAS report programs outside the SDD platform, and load the programs to the report application on the SDD platform.
- SDD does allow you to load your own custom SAS report programs to the report application you build on the SDD platform. They are loaded as code segment objects.
- You can load SAS data sets to the platform to use as input data for the SAS report programs. Or, through the use of the remote generic access engine in SDD v2.0, you report programs can directly access database tables for input.

DEFINING SDD REPORT PROCESSES

SAS report programs are loaded to, or created on, the SDD platform as code segment objects. But SDD does not directly execute SAS programs even in code segment object form. Rather, SDD submits the code to the native SAS System that is resident on the same server that hosts the SDD application. However, SDD cannot submit the code to do this when the code

is in code segment object form. Rather, the programmer-user must first define a SDD report process object from the code segment object. SDD can then submit the SDD report process to execute on the host server platform.

Defining a SDD report process also enables these other important features, as listed here and explained just below:

- Allows the user to identify for SDD each input and output file for the SAS report program (code segment) and its folder location.
- Allows all the output files to be handled in an SDD report bundle object when the report process executes.
- Allows the programmer-user to build the parameter-solicitation screen for the SAS report program.

Identify input and output files: In v2.0, SDD cannot use the libname and filename statements in the SAS report programs (code segments) to identify the input and output files and their locations as is done in the native SAS System. Rather, the user must explicitly identify for SDD each input and output file of a report program. This is done when the report program is converted to an SDD report process. My understanding is that post v2.0, libname and filename statements will work with SDD report processes, and thus this part of the report process definition will change.

Enable report bundle objects: The SDD report process object is designed to place all of the generated output files in a SDD report bundle object. This includes all file types generated, such as Log, List, Data Set, and External Data File. Think of the report bundle object as a single, temporary folder that holds all the files generated. The user can open this folder, and then open each individual item in this folder for viewing. The user can then permanently save the bundle as a whole, or permanently save the individual items in the bundle.

Parameter-solicitation screen: As stated above, SAS report programs are typically parameter-driven and the parameters are typically set by the end-user at run-time. Thus, the report application must allow the programmer-user to define a unique parameter-solicitation screen for each report program. SDD supports this need in an automated fashion as part of the report process definition. This is done using the Process Parameter Editor, as follows:

- First, you load the SDD code segment (i.e., the SAS report program) into the code window of the SDD Process Parameter Editor.
- Next, SDD parses the report program and finds each macro variable in the program. During the parsing, a parameter table is built and displayed in the Process Parameter Editor, where each macro variable found in the program occupies one row in the table. This parameter table is interactive to allow the programmer-user to enter additional information about the macro parameters.
- The programmer-user first indicates, in the table, which of the macro variables found in the program are actually screen parameters to be set by the end-user at runtime. The Process Parameter Editor then focuses further on only the macro variables that are screen parameters.
- The programmer-user then uses other interactive settings in the parameter table to set a variety of attributes for each screen parameter.
- Once this is done, SDD has enough information to generate an HTML file to use as the parameter-solicitation screen that the end-user will later see in the browser window when the SDD report process is executed. Standard GUI objects are used on the screen such as those found in any HTML form. And during this process, SDD provides the link between the HTML form fields on the parameter-solicitation screen and the SAS macro variables in the associated report program.

The entire process just described is menu-driven and is done without the programmer-user writing any HTML code. This is probably one of the most valuable built-in features of the SDD product in terms of using it as the platform for your report application.

Summary points:

- Through its report process object, SDD provides the ability to launch your SAS report programs as native SAS processes on the server hosting SDD.
- The SDD report process object allows you to identify to SDD the input and output files and their locations for your SDD report processes.
- The SDD report process object allows you to handle all the output files generated by your report program in a single report bundle object.
- The process of building a SDD report process also provides a means for the programmer-user to build a custom parameter-solicitation screen for each report program. The screen is an HTML file that displays in the web browser window, and it is built without the programmer-user writing any HTML code.

EXECUTING REPORT PROCESSES

Report process objects are established on the SDD platform by programmer-users. Once they are ready, the end-users can execute them and produce the desired report outputs. I describe here the basic process that an end-user will follow.

The end-user will logon to the SDD platform and be authenticated by SDD. The end-user will navigate the Explorer-like interface to reach the given drug project area on the platform. SDD will control access and authenticate privileges as the user navigates. Once in the drug project area, the end-user will create a new folder object to use for the current reporting session. The end-user will then navigate the folder structure to the folder(s) where the drug project's report process objects are stored,

and make copies of the desired objects in his/her session folder. The end-user will then select one or more of these report process object in the session folder for execution. For each one selected, the end-user will automatically see the parameter setting screen (HTML file displayed as a web page in the client browser), and will proceed to set the parameters and launch the report process for execution.

End-users often want to save the work they have done in a session. That is, they want to save the report programs (i.e., SDD report processes) they have selected and the parameter settings they have made as they executed each report. In the work flow I have just described, the set of report process objects that the end-user assembled in the session folder will be permanently saved because the session folder is a permanent object. And SDD automatically saves the parameter settings made by the user when the report process is opened and executed. Note, though, that although the report processes assembled in the session folder do represent a batch of reports, in SDD v2.0 they cannot be submitted together to execute. They must be submitted singly. I understand that batch submission is planned for a post v2.0 release.

Summary points:

- End-users can select a set of report process objects from among all available, and save a copy of each one in their private session folder. They can then submit any or all of their selections for execution, individually in SDD v2.0, and as a batch post v2.0.
- When each report process object is submitted, the end-users will be prompted to set report parameter values on a web page interface that is customized for the selected report process. Their parameter settings are automatically saved for re-use in a subsequent session.
- The work done in their reporting session to select reports and set parameter values is saved permanently in their session folder.

VIEWING REPORT OUTPUTS

Report process objects can produce one or more output files. Typical output files include Log files, List files, data sets, RTF and PDF files, etc. The report process object is designed to generate its set of output files in a report bundle object and surface the bundle object in a temporary folder location. End-users will proceed to the report bundle folder, open a given bundle, and preview the individual items in the bundle. End-users can move final output files to permanent, project-specific folder locations on the SDD platform, with SDD controlling access and privileges. And SDD provides support for downloading output files from the SDD platform to another platform on the user's network.

SDD provides the capability to open and view most common file types found in a report bundle, such as those just listed above, using the client-based or server-based software applications associated with those file types.

SDD v2.0 does not provide a capability to provide a permanent name and location for a report bundle as the report process executes and the bundle is created. The bundle is given a temporary name and location by SDD and the user has to then permanently save and name the bundle. I understand that an automated permanent save of the bundle as it is being created by SDD is planned for post v2.0.

Summary points:

- SDD provides a report bundle object for handling/managing a report processes' multiple outputs files.
- SDD provides the ability to preview the various report output files using the client-based or server-based software applications associated with those file types.
- The bundle is given a temporary name and location by SDD. The user can preview the output items there, and then save the bundle or its individual items to a permanent location.

SUMMARY

This paper has provided an early look at SDD v2.0 from the perspective of its ability to function as a report management application. In this application, the SAS macro report programs are developed by the user external to the SDD platform on an on-going basis, and then added to the report application environment. The report programs are parameter-driven SAS macro programs.

The focus in this look at SDD has been on how SDD can provide this report application using its standard, out-of-the-box features. It was shown that SDD is both a software application that already provides some of the pieces you need for your report application, and a platform which you can configure to provide some application pieces yourself.

A report management application can be built with the following characteristics:

1. The SAS macro report programs that are developed external to the SDD platform can be loaded to the report application on the SDD platform on an on-going basis.

2. The report programs can be executed by non-programmer end-users who are provided a user interface to select the reports, set the parameter values, execute the reports, and preview the report outputs.
3. The end-users set the parameter values for each report on a web page that is customized for that particular report.
4. The process of building the parameter-solicitation screen is automated within SDD. It is menu-driven and is done by the programmer-user without writing any HTML code. Within this process, among many other things, SDD handles the task of linking the HTML form fields on the screen with the SAS macro variables in the SAS report program. This overall support in SDD for building a custom parameter-solicitation screen for each report is probably one of the most valuable built-in features of the SDD product in terms of using it as the platform for your report application.
5. All output files of a report program are handled in a single bundle that is surfaced to the user. The user can preview the individual bundle items, and save the individual items or the whole bundle to a permanent project location.
6. A full set of administration features are built into the SDD platform and leveraged for the report application. These features cover userid and user group creation and management, and access control through authentication of users both to the overall platform and to individual project folders within the report application on the platform.
7. The report application is 21CFR11-compliant because it is built upon the SDD platform where all objects created (e.g., files and folders) receive the following 21CFR11-compliant treatment:
 - Versioning and auditing of all object creation, modification, and deletion activities.
 - Access to all objects is controlled. Control is based upon single userids and user groups, and assigned roles per group.
8. If the SDD platform is also used by the biostatistics department as the regulatory-compliant repository for all its processing, then the report application's folders can easily integrate into the department's overall regulatory-compliant environment.
9. The report application will use the contemporary architecture of the SDD platform, as follows:
 - It is a server-based, web-enabled application.
 - The client is thin and the user interface is a web browser.
 - SDD can connect to remote databases on other servers to retrieve data for processing on its platform, in an n-tier arrangement.

In addition to the above available features, some desirable report application features identified throughout the paper are not available in SDD v2.0. They are planned for future SDD releases post v2.0.

The SDD product can be used out-of-the-box to build a basic report management application today with many strong features. And the SDD product probably meets about 80% of the needs of a more robust report application. Some of the gap for a robust report application will be filled in future releases of SDD beyond v2.0, based upon expressed plans of the SAS Institute. In addition, the platform's API provides users with knowledge of Java or Javascript the ability to add additional features to their SDD-based report application.

REFERENCES

Helton, Edward D., Halley, Patricia B., & Handlesman, David D. "SAS Drug Development: A Solution for Addressing 21 CFR Part 11 Compliance".
http://support.sas.com/documentation/whitepaper/downloads/50335_0402.pdf

"SAS Drug Development: Intelligent Research Management"
http://support.sas.com/documentation/whitepaper/downloads/50163_0402.pdf

FDA, *21 CFR Part 11*, Electronic Records; Electronic Signatures; Final Rule. Federal Register Vol. 62, No. 54, 13429, March 20, 1997.

ACKNOWLEDGEMENTS

The author would like to thank David Olaley of the SAS Institute staff who gave generously of his time to provide me with information about the coming version 2.0 of SAS Drug Development and to review my thinking.

CONTACT INFORMATION

Barry R. Cohen, President
Planning Data Systems, Inc.
PO Box 666
Ardmore, PA 19003
610 649 8701
Barry.Cohen@PlanningDataSystems.com
SAS Alliance Consulting Partner
SAS Certified Professional

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.