PharmaSUG2010 – Paper DM08

SDD Unleashed - Extending the Capabilities of SAS Drug Development

Stephen Baker | d-Wise Technologies | Raleigh, NC

ABSTRACT

SAS® Drug Development is well known as a compliant, hosted repository for storing SAS data and running programs in a secure and audited environment. What is not well known is that SAS Drug Development provides an Application Programming Interface (API) for extending this environment to be more relevant to the way people do business. Where are the features for automating the upload of transactional data, or for defining a new study? Well, while those features do not exist out of the box with SAS Drug Development they can be obtained through a variety of methods to extend the capabilities of SAS Drug Development. By taking advantage of the remote API and understanding the inner workings of the SAS Drug Development product, custom plug-ins can be developed to improve the business process. Workflow can be introduced, automating manual, time intensive tasks in a way that translates to more time spent on valuable science rather than administering a tool. This paper will discuss use cases where SAS Drug Development can become vastly more powerful by considering adding custom plug-ins to the environment to bring the benefits of workflow and the efficiency of automation to the SAS Drug Development experience.

INTRODUCTION

As described directly in the SAS website, "SAS Drug Development enables the storage and sharing of clinical data throughout an organization via a well-managed, controlled, centralized repository. This single version of the truth enables rapid, efficient sharing of resources, data and knowledge across clinical trials, physical sites, and at all phases of the clinical development process."[1]

Perhaps equally compelling as the features that enable clinical collaboration, SAS Drug Development meets another critical need for the pharmaceutical clients – 21 CFR Part 11 compliance. 21 CFR Part 11 is guidance issued by the Food and Drug Administration (FDA) that stipulates regulatory considerations for, among other things, software systems that are involved in processing electronic data. Consequently, the hosted SAS Drug Development solution provides turn-key compliance for the storage and access of data as well as the execution of programs for viewing, manipulation, and analysis of data brings.

This paper will examine the features of SAS Drug Development that support enhancing its out-of-the-box capabilities and present case studies for how d-Wise leveraged these features to build add-on functionality that mirrors the business processes of clinical organizations to enhance productivity and automate common tasks. To frame this discussion, it is prudent to first review the out-of-the-box features SAS Drug Development provides:

- Secure and customizable repository allowing a business to define a hierarchy, customized objects and metadata, and object level permissions
- SAS engine allowing users to build and run SAS programs within a variety of ways
- Exploration tools to analyze metadata and data within a single data set, across data sets, and across studies
- Compliant environment providing audit trails, versioning and electronic signatures

A full treatment of the features of SAS Drug Development is beyond the scope of this paper, but even in the brief listing above it is immediately obvious how an organization that leverages SAS for data management and analysis can find SAS Drug Development a convincing proposition. While the solution provides a significant amount of functionality out of the box, companies who have implemented SAS Drug Development are often challenged to mate their existing business process exactly to the functionality provided within the product. As with any off the shelf solution, a tool designed for a broad audience cannot be expected to fully address the unique business processes of

every potential customer. Instead, successful products deliver a platform of functionality that addresses the most common needs of their customers. Even if there is a 90 match in the solution and the organization's business processes, the remaining 10 percent can be a source of frustration. As companies attempt to shoe horn their existing processes into the solution's functionality, such gaps may manifest as requiring manual steps where automation would be preferable or as the need to integrate with other tools not readily supported by out-of-the-box features. Many companies implementing or considering SAS Drug Development have seen this same paradigm – the system is going to get them almost the whole way to where they want to be, but what will take them the "last ten yards" that make all the difference between a tool and a solution? The SAS Drug Development integration APIs are one often overlooked out-of-the-box feature that speaks directly to this need for extending this environment to be align with the way organizations and people do business.

Most companies understand how to long into SAS Drug Development via a browser and navigate the interface, and many also utilize the web drive feature to map directories in the solution right onto their users' desktops, but most people don't realize the SAS Drug Development architecture provides many ways to interface with the various components of the system ***without*** using this default functionality. SAS Drug Development provides both a Java and SAS based API that allows you to write code directly against the system and perform functions that would normally be used through the standard SAS Drug Development web based interface. By using the API you can automate functionality that has always been point and click or introduce new functionality to help close gaps in your business processes.

This paper will provide an overview of the capabilities available within SAS Drug Development to add functionality to improve your business process and will provide case studies describing how d-Wise used these capabilities within specific customer projects.

WHAT IS AN API?

API is an acronym for application programming interface, techno-speak for a software component that enables programmatic access to features. To the software developers familiar with Java, an API is a familiar concept. For the SAS programmer, an API might be less familiar. A similar paradigm from the SAS programming world is the SAS Macro. A SAS Macro is a module that can take modularize a given set of functionality, can react to input parameters or return output parameters, and most importantly provide a way to share reusable code between multiple colleagues. With this in mind, think about what might be possible if there were some SAS Macros for interfacing with SAS Drug Development. Time intensive manual tasks such as a bulk upload of metadata tables to a hierarchy of folders could be automated with such a macro using a looping data step. A set of Java APIs could expose this same ability to the community of Java developers.

Much like a UI, or user interface, describes the web pages and widgets that present the features of an application to the user, an API specifies how to touch the application features by writing code. As software designers code systems, they often will build components to model system elements and their respective tasks – for example, a user module that provides services such as creating a user or granting an existing user access to a specific function or a dataset module that understands the rows, columns, and table metadata and allows users to view or update the contents of a table. An API can be public or private, and there are valid reasons for both. For example, it would be beneficial for software developers to build an audit API for a system that has rigid audit requirements and it would be prudent for this API to be private so that system users would not have programmatic access to manipulate the robust audit trail created by the system, potentially compromising its integrity. On the other hand, an API for creating users might be public since system users might find benefits in being able to programmatically create users to mirror users accounts from another system such as LDAP or Active Directory.

OPTIONS FOR EXTENDING SAS DRUG DEVELOPMENT

There are a variety of technical options for extending the capabilities, collectively these are known as the "Remote API". The Remote API comes in two flavors, a Java API and a SAS Macros API. In both cases, the API provides a local or client side view of the SAS Drug Development features and data repository that can be manipulated programmatically. There is a lot of technology in the complete stack that makes this possible, but for the reader all

that is really important to understand is that the APIs are validated and secure and the connections between the code and the hosted SAS Drug Development environment are encrypted and transparent – a long way of saying the APIs abstract away the complicated aspects of programmatically interfacing with SAS Drug Development such as low level transport protocols, authorization, and encryption. Programs you might write using the API go through the very same rigorous login and auditing components of the systems that are behind the web interface you would see in the browser. Under the covers, SAS Drug Development uses webDAV; a web based distributed authoring and versioning protocol, which allows the editing and managing of files collaboratively on a remote web server. Much like the way programs might be coded to understand a file system or database, webDAV provides an interface for content stored in a remote server.

REMOTE API – JAVA INTEGRATION

The Remote API is a java based set of methods that enables developers to extend the functionality SAS Drug Development by providing programmatic access to SAS Drug content, metadata, and specific actions. The API provides java classes and interfaces that a developer to perform actions programmatically such as:

- Connecting and verifying access to objects
- Reading and writing user and object metadata
- Pulling or pushing files to appropriate locations within the repository
- Creating and modifying permissions on objects

The Remote API has two components, the server component and the client component. The server component must be installed on the SAS Drug Development server and can be requested from SAS. The client component lives entirely outside of the SAS Drug Development installation and only requires access to SAS Drug Development via the https connection. It can be installed locally on Windows machine and only requires the Java Development Kit 1.4.2.

The Remote API uses services to make connections to SAS Drug Development and issue methods to perform actions. The service used most often is the repository service which provides methods to work with containers and files within the repository. It provides the ability to read, create, and modify properties on objects, move and copy objects within SAS Drug Development and external file systems, enable versioning on objects, retrieve older versions of objects, and check in and out objects. The security service provides the capability to read and modify permissions on SAS Drug Development objects. The user service provides methods to read and work with user accounts including retrieving user metadata, creating and updating user accounts, and modifying system policies for user accounts. The group service works with user groups including the ability to read user groups, create and modify user groups, and add or remove users from groups.

As mentioned earlier, the Remote API is a java API and all that is needed to use it is a SAS Drug Development user name and password. The first step is to make a connection to SAS Drug Development by creating a session. A session stores the state of the connection to the server and provides the access point to the other services described above. Below is an example of a method implemented to obtain this initial session.

```
public static Session getSession() throws SDDFileAdapterException {
      ApplicationContext appContext = ContextManager.getContext(CONFIG_FILE);
      String url = (String)appContext.getBean(SDD_URL_BEAN);
      String userId = (String)appContext.getBean(SDD_USERID_BEAN);
      String password = (String)appContext.getBean(SDD_PASSWORD_BEAN);

      log.debug(MessageFormat.format("Connecting to {0}, using {1}\\{2}", new Object
[] {url, userId,
    password.replaceAll(".", "*")}));
      try {return SessionFactory.newSession(new URL(url), new
UsernamePasswordCredentials(userId,
    password.toCharArray()));
      }
```

```
      catch (Exception e){
         throw new SDDFileAdapterException("Could not create session to remote
server.", e);
      }
   }
```

Once the session is created it can be used to access the Repository Service and begin working with objects within SAS Drug Development.

REMOTE API – SAS MACROS

In addition to the Remote API, there is another option available to developers to interface with SAS Drug Development. The SAS Drug Development API Macros are a set of SAS macros that provide familiar syntax to users and provide similar capabilities as the Remote API. While the macros don't provide the same breadth of functionality as the Remote API, the familiarity of SAS code provides an easy learning curve for the typical user interacting with SAS Drug Development. While the SAS Drug Development API Macros use SAS code on the surface, under the covers the macros use the JAVAOBJ capability introduced in SAS 9.1 to make calls to the Remote API described in the previous section. Therefore, the macros become an interface to the Remote API.

The SAS Drug Development API Macros require SAS 9.1.3, the JRE 1.4.2_XX (but not 1.5), and the appropriate Remote API client component of the SAS Drug Development version your company is using. After a few minor configuration of the SAS environment as well as the system Java options, the macros can used within SAS 9.1.3. The code contains over 35 macros to interact with the SAS Drug Development system. To start using the macro only requires a few simple steps. The first step is to login to SAS Drug Development using the following macro:

```
%sasdrugdev_login(url=%str(https://sddinstance.sas.com/sddremote/),sdduserid=%str(cd
ecker),sddpassword=%str(**))
```

Once the user is logged on, they can use any of the other macros to work with objects within SAS Drug Development. Once the program finishes, the user should use the following macro to log off of SAS Drug Development cleaning up the session.

```
%sasdrugdev_logout
```

The SAS Drug Development API macros provide a variety of capabilities. Below is a sample of some of the most used macros including the following:

- SASDRUGDEV_GETOBJECTS – Returns a list of all objects and associated metadata within a specific SAS Drug Development path
- SASDRUGDEV_CREATESDDCONTAINER – Creates a folder within SAS Drug Development based on SAS Drug Development path
- SASDRUGDEV_SETCONTAINERPROPERTY – Allows a user to modify a property on a folder once it is created
- SASDRUGDEV_SETUSERPROPERTY – Allows a user to set the properties on a user including new passwords

The SAS Drug Development API macros provide users the ability to create a wide range of utilities for working with the objects in SAS Drug Development and automating their business processes.

CASE STUDIES – EXTENDING SAS DRUG DEVELOPMENT IN ACTION

Up to this point, this paper has focused on the concept of the SAS Drug Development remote Java API and SAS Macro APIs. A User's Guides for each API is available for SAS Drug Development version 3.4 and 3.5 (see the "recommended reading" section at the end of this paper). These documents cover the APIs exhaustively and should be referenced for more information about what is possible with this feature of SAS Drug Development. Keep in mind

that your installation of SAS Drug Development may or may not already have the APIs available, contact your SAS representative, organization program manager, or d-Wise to discuss whether adding these features is appropriate and how your objectives might be possible with the remote API in place. The next section or the paper reviews several case studies where the APIs were leveraged to make build features that closely reflect tasks that clinical organizations needed to accomplish that were not supported by out-of-the-box features of SAS Drug Development.

CASE STUDY 1: CUSTOMIZED STUDY CREATION AND PERMISSION REPORTING

PROBLEM

During the implementation of SAS Drug Development, the customer identified a common task that had no direct answer in the web interface. When starting a new study, a standard directory structure needed to be defined and permissions for various folders in the hierarchy would be assigned to various users. In this new directory structure, certain generic study files, metadata tables, and programs would be added. While this task can be accomplished through a series of point-and-click actions throughout the web interface, the task seemed appropriate for automation. Beyond simple time savings, the customer saw value in knowing that an automated process would ensure the common folder structure and necessary permissions were applied in a consistent manner. Automating this process could mean time savings and reliability, but would it be possible?

On further investigation, the customer began to wonder whether their existing studies which had been created manually were adhering to their security standards. SAS Drug Development has widgets in the web interface for viewing permissions, but it can be challenging to compare how permissions are set on two directories – for example, viewing user permissions for the "analysis" folder in two different study folders is challenging since you need to open two browser windows and look folder by folder to get all the required information. To date, the customer had been writing SAS code to generate a dataset of user permissions and then wrote reports against this dataset to compare settings. For studies set-up with the new automated wizard permissions could be easily defined and reliably controlled, but for legacy studies already in SAS Drug Development how could the permissions review process be easier and more accessible to users who might not be savvy enough to write or manage the SAS code used today. Would it be possible to build a single interface to review permissions on multiple folder hierarchies for multiple users all at once?

SOLUTION

d-Wise designed an extendible framework including a new user interface using a technology called Adobe Flex. Flex is a software development kit released by Adobe Systems for building applications that are browser and platform independent and provide a variety of ready-made components for rapidly building a rich user experience. The new application sat next to the SAS Drug Development environment, allowed users to login using their SAS Drug Development credentials, and used the java based Remote API to implement all components.

The pictures below show the initial interface for the Study Setup Utility. Within this interface users were able to give the study a name and then navigate to a location within SAS Drug Development. The utility used the Repository Service within the Remote API to extract the SAS Drug Development hierarchy. The users only see folders they are allowed to access. Once users selected the study they were then able to assign users and groups to the specific studies. Behind the scenes they had defined a standard permission model for every study. The underlying code of the utility implemented this model based on the user roles selected on the screen below. After users made all the selections, the user could then select the Create Study button and a new study was implemented and all appropriate permissions assigned.

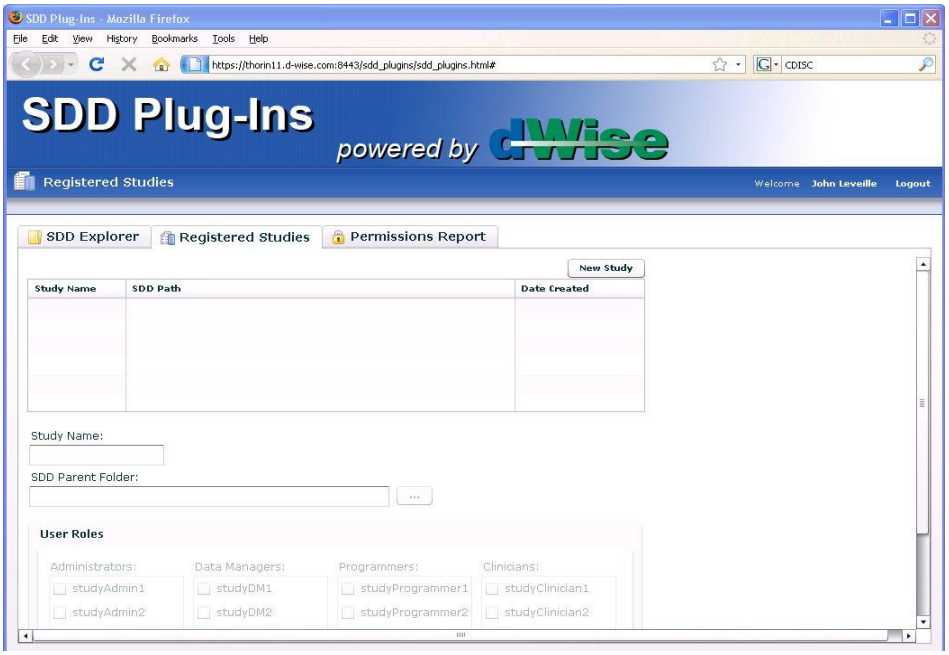

Figure 1. New Study Definition Wizard – A user enters a name for the study folder and browses the SAS Drug Development repository hierarchy to determine the location the new folder should be created.

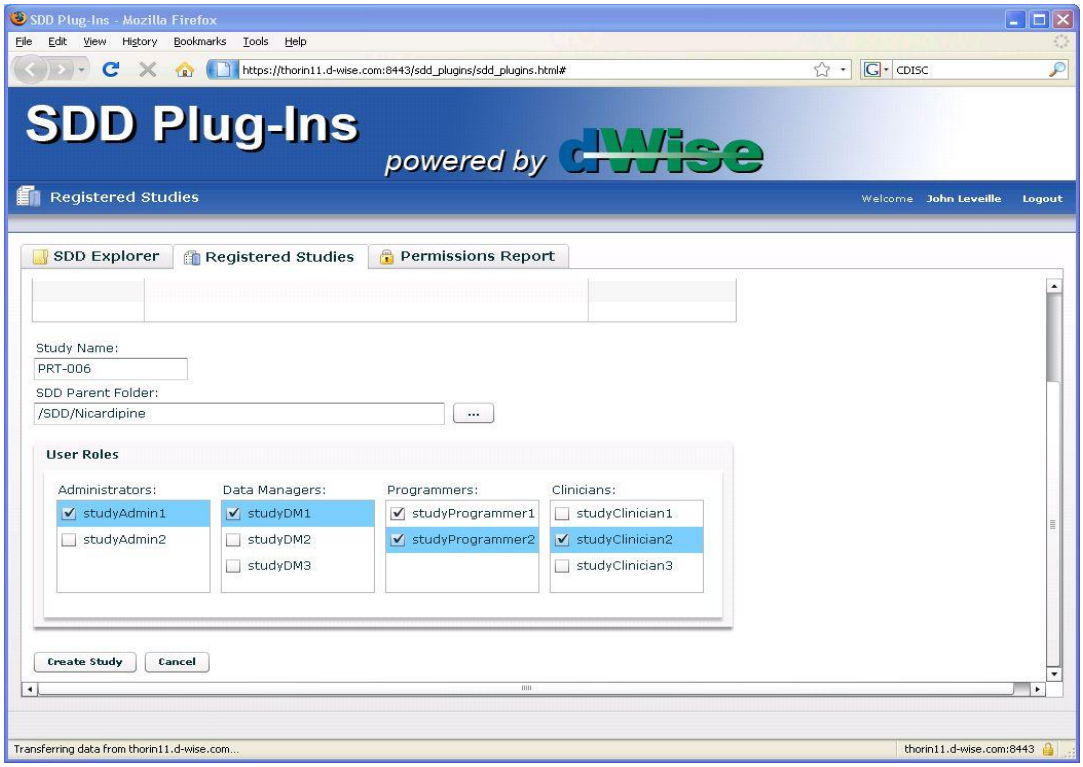

Figure 2. New Study Definition Wizard – The wizard presents the security concepts defined by the organization, such as administrators and clinicians, and the users identifies users for each role.

A second utility implemented was a dynamic permissions report. Users had always requested the need to better understand who had access to which components within a study. Based on these requirements d-Wise implemented an easy to use dynamic permissions report that let users select certain levels and users. The report would then show the permissions from that point downward in the hierarchy with clear delineation of the type of access. In the picture below users can select users and/or groups and the path of interest within SAS Drug Development. Once they have made their selections, they run the report. The report displays the permissions in both text and colors for each of the users assigned to the study and also allowed the user to expand the folders within the study to view permissions at a lower level. Green was used to indicate ALL permissions, partial permissions were displayed in blue and called out specifically (e.g. (R) indicating read permissions), and red indicated no access of any kind.

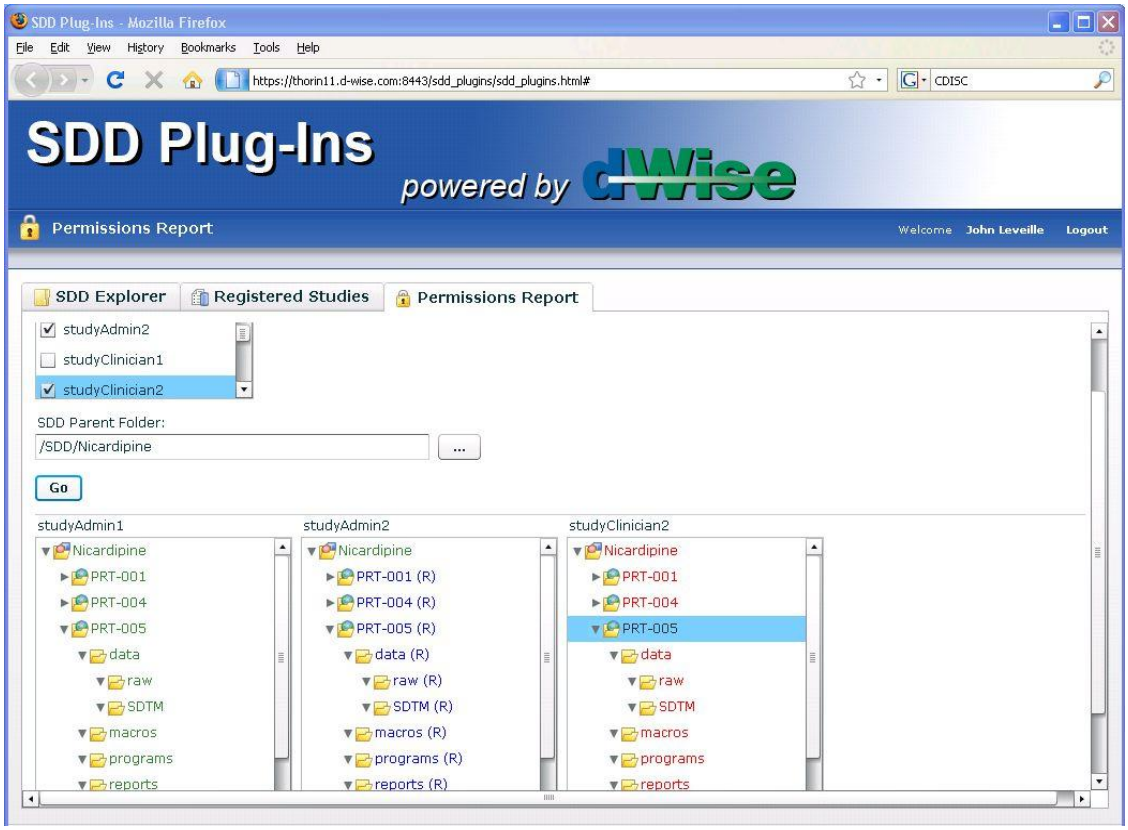

Figure 3. Permissions Report – The permissions assigned to users "studyAdmin1", "studyAdmin2", and "studyClinican2" for the folder "/SDD/Nicardipine" are displayed in a single interface, easily identifying how the three users are provisioned to access (or not access) the folders within the study directories.

BUSINESS EFFICIENCE

By using the capabilities to extend SAS Drug Development, the customer generated an extendable framework which provided the additional tools to support their business process. In this case the Study Setup tool let the customer setup a study in seconds versus a manual and tedious process and the permissions report gave them the easy view and met the requirements of their internal validation requirements. In addition, by implementing an extensible framework, the customer can continue to add value added utilities as the business process requires.

CASE STUDY 2: AUTOMATING INCREMENTAL DATA UPLOADS

PROBLEM

The customer's eClinical environment included not only SAS Drug Development, but also a clinical data repository (CDR) within the organization where data was loaded from an electronic data capture (EDC) application. Once data was available from the CDR from the EDC, the clinical programmers would need to upload the data from the in-house CDR to the remote SAS Drug Development solution. The dual-solution approach here allowed that the CDR store data in a model that made sense for the case report forms understood by the EDC but also to provide extracts in the CDISC Study Data Tabulation Model (SDTM) standard which was commonly understood by the clinical programmers who are less familiar with the nuances of case report forms.

The size of the data in the CDR for a new study might start out small, but over time grew quite large. Clinical programmers needed to start their work before all study data was completed, so there was a need to regularly update SAS Drug Development with the new records from the CDR. Since the upload of data from the in-house system to the remote system requires executing a complicated extract from the CDR and then sending data over the internet to SAS Drug Development, the duration of the extract scaled with the size of the data – meaning as the data grew the upload process took longer and longer.

In an ideal world, the clinical programmers would log into SAS Drug Development and click a button to request new data for their studies and the system would reach out to the CDR and quickly upload the new data to SAS Drug Development so the clinical programmers could get down to business. SAS Drug Development does not provide this type of systems integration of connectivity out-of-the-box, but is the ideal world something possible with the remote API?

SOLUTION

The solution required two parts, components local within the company's infrastructure near the CDR and remote on the servers hosting SAS Drug Development. These two sets of components communicated via web services that provided a wrapper to the Remote API functionality. Within SAS Drug Development, a clinical programmer would browser to a job for their specific study to request new data. The request for new data could be restricted to specific SDTM domains, such as AE or SUPPQUAL, and could be provisioned to request data for an ongoing study or from a point in time. When the user launched the job to run the request, the custom components hosted in the SAS Drug Development environment sent a request over web services to the components on the customer's servers to initiate an extract from SAS Drug Development. The extract examined the request parameters to pull only the incremental updates from the CDR, allowing that the smallest possible data package to be sent over the internet greatly expediting processing time and reducing the demand on server resources. Once the CDR extraction was available, the incremental data was compressed and sent via web services to the custom components on the SAS Drug Development server. The remote components then extracted the compressed archive and merged the incremental updates into the existing data in the SAS Drug Development environment. When the upload was complete and the new data was available, the component used the notification API to send emails to the users that new data was available for their use. If there was a breakdown at any point in the process, the same notification API was used to advise the users and also support staff that an extract was failing and required attention. Beyond just automating and speeding up the data upload process, the support teams were engaged automatically and more quickly on problem extracts so the problem data or system issues could be troubleshoot quickly to keep the clinical programmers on schedule.

BUSINESS EFFICIENCES

Moving data between two remote systems can be complicated and time consuming. Rather than manually extracting data and uploading files one at a time, with this solution provided an automation to uploading new data and simplified the entire process to running a single job in SAS Drug Development. This simplification meant that the clinical programmers no longer needed upload multiple large files to SAS Drug Development manually, nor did they need to be trained to understand how to perform extracts from the CDR. With a reliable framework in place, the focus was placed back where it belongs – not on getting data from A to B, but on clinical programming in SAS Drug

Development with current data. The support burdens and training needs to connect the two systems were greatly reduced and the process of getting data from the CDR into SAS Drug Development was greatly expedited, meaning less time was spent on getting data and more time was available for doing something with it.

CASE STUDY 3: INTERFACE BETWEEN SOLUTIONS

PROBLEM

In this case study the customer had implemented multiple solutions including an independent data integration (or ETL) solution alongside SAS Drug Development. In addition, the customer collected their metadata within a standard Excel template and stored the contents within SAS Drug Development. The customer wanted a mechanism for extracting the files from SAS Drug Development and reading the metadata into the ETL solution. In addition, they wanted the ability to read metadata and data within the ETL solution, build various components such as data table shells, metadata, and data dictionaries, and store all those contents within SAS Drug Development. The end goal was to let users of the ETL solution quickly load the metadata and datasets they needed from SAS Drug Development into the ETL desktop environment and also write changes back into the SAS Drug Development hosted repository.

SOLUTION

While the custom solution involved a number of technologies, the focus described here will include the integration with SAS Drug Development. The first step in the solution was to develop code that would extract the required files from SAS Drug Development. Since the environment was primarily SAS and the customer users wanted the ability to maintain the solution moving forward, d-Wise implemented the code using the SAS API macros.

The code below was used to read files from SAS Drug Development and transfer them to the local environment.

```
**** Login to SAS Drug Development using authorized credentials.  Also using the
debug level for finding errors ****
%SASDRUGDEV_LOGIN(url=%str(URL/sddremote/),sdduserid=%str(user),sddpassword=%str(pas
s),debuglevel=DEBUG);

**** Verify that the Excel file requested exists in SDD ****
%SASDRUGDEV_FILEEXISTS(SDDPATH=SDD Path);

**** Create a file local to the ETL tool.  This macro copies the file from SAS Drug
Development to the local system ****
%SASDRUGDEV_CREATELOCALFILE(LOCALPATH=Local System Path, SDDPATH=SDD Path);
```

In addition to moving files from SAS Drug Development to the local system, the solution had to also process and move files to the SAS Drug Development environment. This was a similar process but used a few different macros.

```
**** Login to SDD using authorized credentials.  Also using the debug level for
finding errors ****
%sasdrugdev_login(url=%str(URL/sddremote/),sdduserid=%str(user),sddpassword=%str(pas
s),debuglevel=DEBUG);

**** Check to see if the file exists.  If it does remove it first ****
 %SASDRUGDEV_FILEEXISTS(SDDPATH=SDD Path);
       %if &_sddrc_=1 %then %do;
          ** Delete SDD file before writing;
          %SASDRUGDEV_DELETEFILE(SDDPATH=SDD Path);
       %end;

**** Copy file from local system to SDD ****
```

```
%SASDRUGDEV_CREATESDDFILE(LOCALPATH=Local System Path, SDDPATH= SDD Path,
type=document);
```

BUSINESS EFFICIENCES

By implementing this solution the customer was able to use the set of different tools familiar to their user but still automate the process. Since the ETL solution and the SAS Drug Development product did not provide ready out-of-the-box integration, it was necessary to write macros to automate the push and pull of data and metadata between the two tools. Since this process impacted a large number of users, having a common component that programmatically automated the integration ensured that all users were accessing and creating data and metadata using the same tools – eliminating the need for each ETL developer in the team to figure out a method for connecting to the SAS Drug Development repository and instead leverage the tool now available explicitly for this process.

SUMMARY

The case studies show that with a little programming gumption, the SAS Drug Development remote API is a powerful tool for unleashing the power of the SAS Drug Development solution into the enterprise. Automating common tasks, building interfaces for importing and exporting data, and enabling systems integration between remote and local solutions are all possibilities demonstrated by these case studies – but the possibilities are limited only by the imagination. Building workflows to help users navigate through business processes or building components to automate standard error checking against new data are other possibilities that d-Wise has seen customers realize by applying the remote API. For more information on the technical details of the remote API see the referenced documentation in the "recommended reading" section below. Think about the repetitive or manual tasks that could be automated to make using SAS Drug Development at your organization more efficient. Custom add-ins like the ones described in this paper built on the Remote API can deliver help clinical organizations go the "last ten yards" to align SAS Drug Development with their business processes and realize efficiencies and reliability that translate to getting things done faster. SAS Drug Development is a powerful tool out-of-the-box - with the remote API a little out-of-the-box thinking can go a long way to making it even more powerful.

RECOMMENDED READING

- SAS Drug Development 3.4 User's Guides
  - Remote API http://ftp.sas.com/techsup/download/hotfix/drugdev/34drgrmtapi03/RemoteAPI.pdf
  - Macros http://ftp.sas.com/techsup/download/hotfix/drugdev/34drgmacro01/SDD_Macros.pdf
- SAS Drug Development 3.5 User's Guides
  - Remote API http://ftp.sas.com/techsup/download/hotfix/drugdev/35drgrmtapi01/RemoteAPI.pdf
  - Macro http://ftp.sas.com/techsup/download/hotfix/drugdev/35drgmacro01/SDD_Macros.pdf

REFERENCES

[1] "SAS Drug Development | SAS". SAS Corporate Website. SAS Institute, Inc. March-27, 2010. http://www.sas.com/industry/pharma/develop/sdd.html.

CONTACT INFORMATION

Stephen Baker, Manager, Strategy and Business Development
d-Wise Technologies
Work Phone: 919-600-6237
E-mail: sbaker@d-wise.com
Web: www.d-wise.com