PharmaSUG 2014 - Paper AD24

Moving to SAS Drug Development 4

Magnus Mengelbier, Limelogic AB, Malmö, Sweden

ABSTRACT

Life Science organisations have a long investment into business processes, standards and conventions that make it difficult to simply turn to a new generation of analysis environments. SAS Drug Development 4 integrates many key features found in current analysis environments that are spread across several applications and systems, which require to be monitored and managed accordingly. The paper considers a set of SAS programs and how the SAS Drug Development repository, workspace and workflow features support a common innovative business process with all of the associated tools and utilities. The result is a short list of points to consider and some tricks for moving a business process from PC SAS or SAS server environments to the new release of SAS Drug Development.

INTRODUCTION

SAS® Drug Development 4 is a significant update to SAS Drug Development 3.5 that has become appreciably more user friendly and useful to an organisation. Whether an organisation is migrating from SAS Drug Development 3.5 or moving to SAS Drug Development 4 from a SAS workstation or server environment, there is the expected change to process, standards and conventions along with a degree of uncertainty.

SAS Drug Development is quite comprehensive and to consider each feature and how it applies is well beyond the scope of one paper. We shall focus on the user perspective, highlighting features and any caveats and issues that may or may not be applicable to provide an introduction.

The SAS Drug Development solution is, in a very generic point of view, foremost a repository with the ability to execute SAS programs. A principal feature is the central repository and the associated workspace. Additional enhancements to the SAS programming development and execution environment have significantly simplified setting up and running programs.

A SAS program with SAS Drug Development is not that different from one used with a SAS Windows environment, either on a PC or workstation and servers. There are some specifics for SAS Drug Development that are a requirement or provide access to resources, e.g. the SAS program becomes aware of SAS Drug Development and its content.

Although the paper does refer to SAS Drug Development macros and the Java API, we shall not discuss them in depth beyond references where it may or may not be useful.

The aim is to provide an overview of how the use of SAS Drug Development may impact business processes and SAS programming. SAS Drug Development does contain all of the major features we expect from an analysis environment. It is only to consider the impact and how your business process and SAS programming needs to change based on your requirements to adopt SAS Drug Development.

SIMPLE ANALYSIS PROCESS

There are countless of approaches to use SAS for analysis, from the open and flexible to rigid process flows, different levels of adopting standards, different interpretation of standards, legacy business processes and so forth. These are a wide array of requirements for any solution to be expected to fulfil. The Life Science industry does however revolve around a consistent set of general business process conventions and outputs.

SAS is used widely to process collected information in preparation for and for the actual analysis, starting from captured and extracted data to final report outputs (Figure 1). This includes source and analysis data sets as well as Table, Listing, Figures and Graphs included as part of the appendix or in the text of statistical and clinical

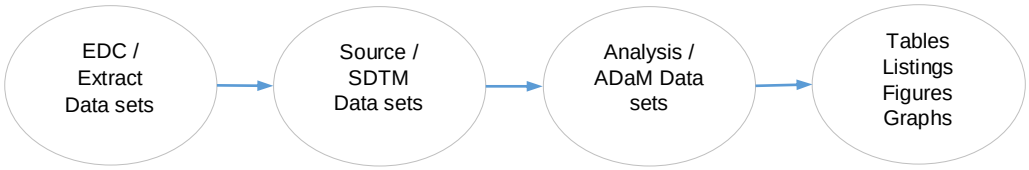

Figure 1. General process convention

reports. Depending on your practice, the result is reasonably similar, e.g. data sets and documents for outputs, with technical implementations dictated by your internal data standards, conventions and processes.

The exact details is beyond the scope of this paper as it will vary from organisation to organisation, but important is that they do rely on SAS programs organised per project, study, pooling, analysis, etc. and that through this hierarchy exists a collection of macro libraries of varying scope, for example a global macro library compared to the local study library.

We shall assume a general approach to SAS programs (Figure 2) that is consistent with performing analysis. Following our example, SAS programs will most often require some form of input (ADSL, AE and EX) and SAS procedures and statements (SAS Code) to produce an output (ADAE), no surprises or special requirements there. As the analysis environment matures, so does the standard libraries, both on the level of a project, study and analysis as well as global or libraries for broader use.

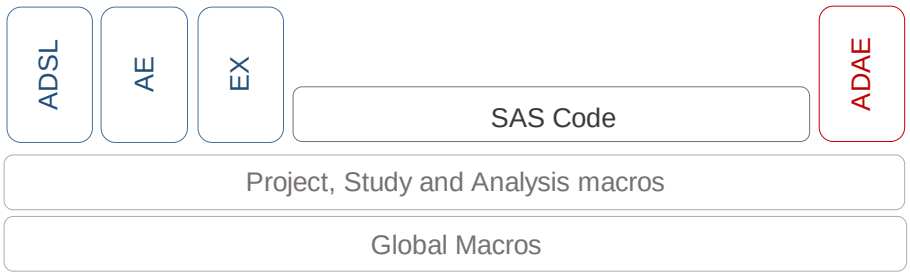

Figure 2. General approach to analysis programs

Another important aspect is that users are most often not expected to manage the library logistics for each program, instead one or more initiation macros will correctly configure SAS macro options to ensure that the correct libraries are available and accessible from within the SAS session, something that we will recognize further on as an important consideration.

Note that the process as described is not dependent on CDISC standards, which is an increasingly frequent requirement for other analysis environments. To SAS Drug Development, CDISC is a business standard and convention allowing for a large degree of freedom in interpretation.

We employ this generic SAS process to discuss the different elements of SAS Drug Development 4, from storing your inputs, programs and outputs to executing SAS code and the implications of migrating from SAS on a local PC, a SAS server and the previous generation of SAS Drug Development.

SAS DRUG DEVELOPMENT 4

SAS Drug Development 4 is the next generation of SAS Drug Development that is essentially a completely new environment, although some internal components are shared with its predecessor SAS Drug Development 3.x.

The current version is SAS Drug Development 4.4, although there are essentially no major differences in application features compared to 4.3.x beyond minor updates, fixes and a change in the colour scheme. SAS Drug Development 4.4 is also based on SAS 9.4.

SAS Drug Development 4 is also comparable to most other analysis environments based on SAS. It provides a central repository for shared storage, an editor to create, modify and manage SAS programs, and a SAS server to execute SAS code.

All features and facilities are integrated in a single environment behind the Web browser interface that is based on Adobe Flash and runs in your Web browser (Figure 3), a so called Single Page Web Application simply because everything you do occurs within the same Web browser window.

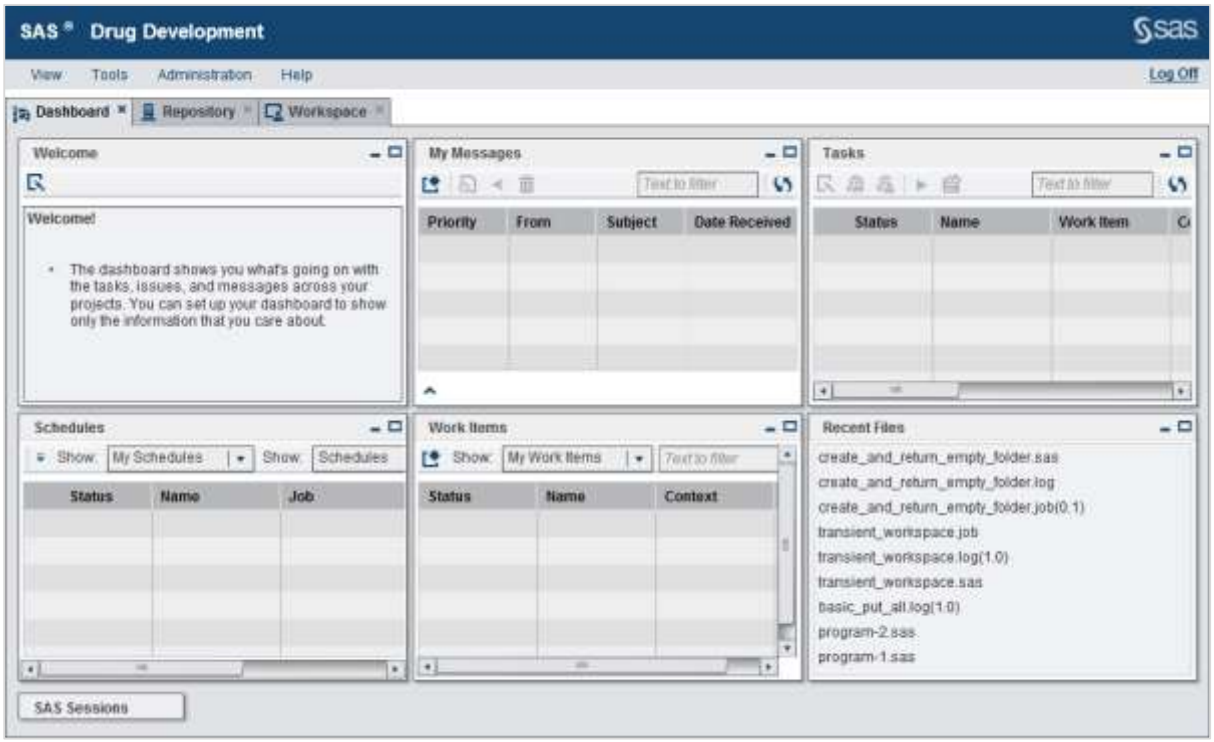

Figure 3. SAS Drug Development 4.4 Dashboard

There is of course additional technical detail, but those specifics are well hidden from the user and are only encountered when discussing the technical merits, scalability and configuration of the solution. As any moderate complex analysis environment, configuring SAS Drug Development is much like arranging a symphony orchestra to perform very different works and, although fascinating, well beyond the scope of this paper.

As we continue, we refer to SAS Drug Development 4 or 4.4 as simply SAS Drug Development.

CENTRAL REPOSITORY AND THE PERSONAL WORKSPACE

SAS Drug Development contains two principal areas to store content, e.g. your inputs, outputs, programs, documents, etc., and from which SAS programs execute, the central Repository and the Workspace. Both are entirely focused on storing content, whether it is an input for or output generated by a SAS program, or simply retained as part of the project or analysis archive.

There are many different ways to compare and describe the two. The central Repository is for most organizations the central source while the Workspace is treated as a personal working copy. One important differentiating aspect is that the Repository has specific rules that essentially ensure content is read-only. You are able to execute SAS programs and store the results from within the Repository, but not edit content directly which also extends to SAS programs. The Workspace is by design and intent the natural place where a user performs most of the exploratory analysis and program development.

CENTRAL REPOSITORY

The Repository is a simple file system based on folders and files, similar to a Windows network share and a Unix/Linux mount, which is extended with additional application features and attributes. As we shall discover, the file system principle is consistent with most SAS environments and makes a transition to SAS Drug Development easier as the impact on an individual SAS program is less.

The Organization, Project and Analysis

A non-trivial application feature in the SAS Drug Development design is the Organization, Project and Analysis container, which is the cornerstone of the SAS Drug Development directory structure. A container is a regular folder with additional application metadata that is used to manage access, privilege, permission and tasks, the latter considered in further detail when exploring SAS Drug Development workflows.

The Organization container is always the root and is the parent of all content in the repository. If SAS Drug Development would become multi-tenant at some point in the future, the Organization container is a natural

extension to contain all the repository content for a specific tenant, or organization. In the current single tenant environments, it is the Repository root folder.

The additional Project and Analysis containers are the next two levels in the path, e.g. subfolders or sub-containers to the Organization to be precise.

For example, a simple path would consist of the following containers.

Organization *Project* *Analysis*

/ QAGPharma / PharmaSUG_2014 / PharmaSUG_2014_Analysis

The position in the path of the containers, including Project and Analysis, is currently not configurable and cannot be separated by folders, which may restrict adoption by organizations that use a different directory structure and convention.

Each container contains a dedicated location to store folders and files, the *Files* folder, associated with each specific container (Figure 4). The Files folder in the container is required and added by default, which implies that all folders and files added to PharmaSUG_2014_Analysis in our above example would be stored in the folder `/QAGPharma/PharmaSUG_2014/PharmaSUG_2014_Analysis/Files`.

As it is still a simple file system in the end, it is possible to read, create, modify and delete folders and files in other containers, provided you have the appropriate permission.

So, where is my Study area? Defining an additional container is currently not possible, but we are allowed poetic license. One possible approach is to equate the third level, Analysis, to Study Analysis as containers currently do not have any function beyond defining additional metadata.

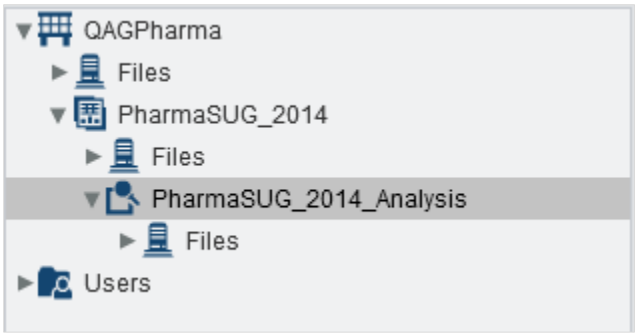

Figure 4. Basic folder structure

Our previous example then becomes,

Organization *Project* *Analysis*

/ QAGPharma / PharmaSUG_2014 / PharmaSUG_2014_Study

The different study analysis planned or performed, such as Interim 1, Interim 2, Final Dry-Run, Final, etc., are then the first folder level within the study Analysis container.

Recycle Bin

One additional feature of the Repository to highlight is the Recycle Bin, since it plays an important role with content and its versions, signatures and attributes. The Recycle Bin captures any folders and files deleted from the Repository.

All versions, electronic signatures, permission, etc. associated with a folder or file being deleted are retained together with the folder or file in the Recycle Bin.

A deleted folder or file can be restored from the Recycle Bin to its original location. The original permission, versions, signatures, etc., are restored along with the folder or file, which implies that permission is not updated based on the default permission defined for the parent folder. A user restoring a folder or file has to ensure that the restored folder and/or file permission is correct as permission may have changed in the Repository while the folder or file was residing in the Recycle Bin.

Deleting or restoring only a specific version of a file is currently not possible. The operation is performed on the entire repository item.

Once a folder or a file is deleted from the Recycle Bin, it is permanent in nature. Relying on the environment back-up and restore process is near impossible as there are multiple background services and technical constraints that have to be manually updated.

Defining a strategy for managing the Recycle Bin is important as the performance of certain SAS Drug Development features is impacted as the Recycle Bin size increases. A recommendation as to an acceptable number of items in the Recycle Bin is difficult to provide as it is dependent on different factors, such as

environment configuration, the structure of the directory tree, number of users, groups and complexity of the permission scheme, just to name a few.

Concept of Versions

The Repository contains version control capabilities for files, but not folders. As the application does not contain a system-wide option to enable version control for all files by default, version control must be enabled and managed for each file on a case-by-case basis.

The initial version of a file can be set as minor (0.1), major (1.0) or custom (*major.minor*). As noted in the parenthesis, identifying a version in SAS Drug Development supports the concepts of major and minor revisions.

A user is provided the choice of the next version when updating or modifying a file under version control. Minor and major revision increments the corresponding number by one. A user may also assign a custom version number, as long as the new custom version is a *later* version than the current version. This rule does not apply if the Repository file is not under version control.

Version control can be disabled on a file under version control, which would retain the latest version and all prior revisions discarded. Unfortunately, re-enabling version control on a file does not restore prior versions or start incrementing at the last assigned version number, the latter if the file was previously under version control. The discarded prior versions are also not available in the Recycle Bin.

The combined possibility to disable and discard prior versions, re-enabling version control with a custom version number and that the application does not enforce persistent version identifiers may be an issue, if a critical, regulatory or compliance document refers to a specific version of a file. In this scenario, it is possible to replace the content of a file while retaining the version label, whether intentional to fix an issue or unintentional. The only indicators would be last modification date and the audit trail, which would need to be analysed when asserting the file integrity. It is therefore a good recommendation to use more than the version label to identify a specific version of a file.

The role of version control in SAS Drug Development is further discussed as we consider the interaction between the Repository and Workspace.

Electronic Signatures

SAS Drug Development supports electronic signatures for files, but not containers or folders. If you wish to sign a Project or Analysis container or a specific folder, you would sign all files by electing to sign the container or folder.

An electronic signature in SAS Drug Development 4 consists of the role (who you are signing as), the reason (why you are signing), an optional comment and the user credentials. Signing a file requires the user to re-authenticate with a valid password. The certificate based signature in SAS Drug Development 3.x is no longer supported.

Not all users have default privilege to sign files. It is granted on an individual or group basis using container roles that are briefly mentioned when considering privilege and permission. It is however important to note that signature privilege cannot be assigned for a specific type of file or context, e.g. you either have or have not been granted the privilege. For example, a Data Manager cannot be granted to only sign files of type data sets in the DM study analysis folder.

Many organizations electing to use electronic signatures in SAS Drug Development do so for business process compliance, for example to support 21 CFR Part 11 and/or remove dependency on hard copy records. There are no current features in SAS Drug Development that depend on or require the signature feature to function.

WORKSPACE

The other location to store content is the Workspace, which is a personal user area independent of the Repository much like the relationship between the C-drive on your local laptop or workstation and the network file system. This implies, for example, that Repository content is only available in the Workspace, if it was copied or moved there.

Each active user is provided a personal Workspace and it is only accessible by that individual user. SAS Drug Development (up to including version 4.4) does not contain a privilege that will grant a user access to other users Workspaces, even for administrators.

The distinction that each *active* user is provided a Workspace is important as a user's Workspace with all its content is permanently deleted from the environment when the user account is set inactive. The current process does not produce or create a back-up or archive copy.

The Workspace file system is also primitive as it only understands folders and files. Additional features such as version control or those added to the Organization, Project and Analysis containers are neither not available nor deemed necessary. As such, a container defaults to its simplest form, a regular folder, in the Workspace.

Another feature not available to the Workspace is a Recycle Bin. Once a folder and file is deleted, it is permanent.

The list of caveats, only some listed here, does not mean that the Workspace is less useful. The basic nature will greatly simplify how SAS programs are developed, managed and interact with content stored with SAS Drug Development.

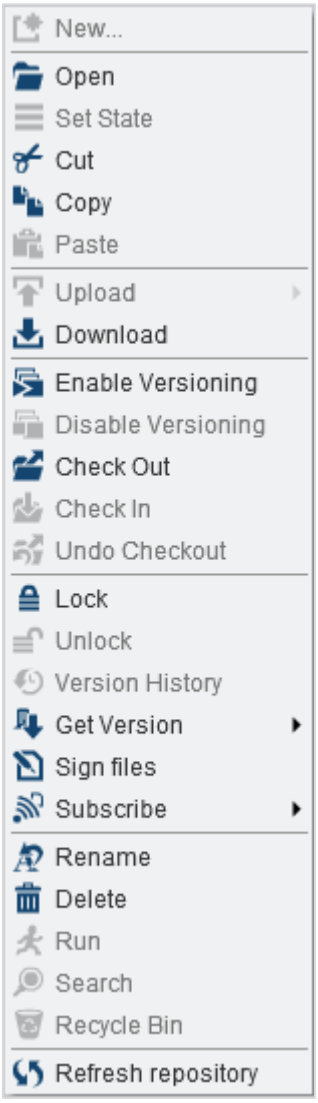

Figure 5. Example Repository menu for a file

WORKING WITH THE REPOSITORY AND WORKSPACE

Even though the Repository and Workspace are two independent features of SAS Drug Development, they are designed to be an integral part to complete most tasks and activities in SAS Drug Development.

The Repository and Workspace is access through the primary user interface, which also includes moving and synchronising content between the two. Currently, there is no feature to allow linking content in the Repository from the Workspace or establishing some form of automated synchronisation. All tasks are manual and performed through standard pop-up menus or the toolbar.

As a rapid overview of key Repository options (Figure 5), a user is presented with several actions provided the user is granted appropriate permission. All options are visible with those options not applicable nor granted being inaccessible.

The options *Cut*, *Copy* and *Paste* are similar to the actions in a Windows environment. In SAS Drug Development they are all performed either in the Repository or Workspace. *Cut*/*Copy* and *Paste* from the Repository to Workspace, or the reverse direction, is not possible.

A user is able to enable or disable version control (*Enable*/*Disable Versioning*) for one or more files directly through the pop-up menu. The user is prompted to specify the initial version for each individual file in the selection as version control is enabled.

The feature to transfer content from the Repository to the Workspace is delegated to either *Get Version* or *Check-out*, both used regardless if version control is enabled or disabled.

Get Version will allow you to copy a folder, including all subfolder and files, or a file to the Workspace. *Get Version* actually is performed as *Get Latest Version* or *Get Specific Version*, as you require. If the file is under version control and a prior version is required, select a specific version with *Get Specific Version*.

The option *Check Out* will copy the latest version of a file to the Workspace and lock it for editing. The lock remains until you *Undo Checkout* or *Check In* the file, possibly with changes. If a user does *Check In* a file under version control, the user will be prompted to provide details that will increment the version label appropriately.

The options *Check Out*, *Undo Checkout* and *Check In* are only applicable to files and thus cannot process empty folders.

All of the above menu options that transfer content between the Repository and Workspace will assume, and create if necessary, the same directory structure in the Workspace and Repository, depending on the direction of the requested action.

The Workspace also has a specific menu (Figure 6), although much simplified. The one item to mention is *Mark for Addition*, which will ensure that the file is in a path that is or can exist in the Repository.

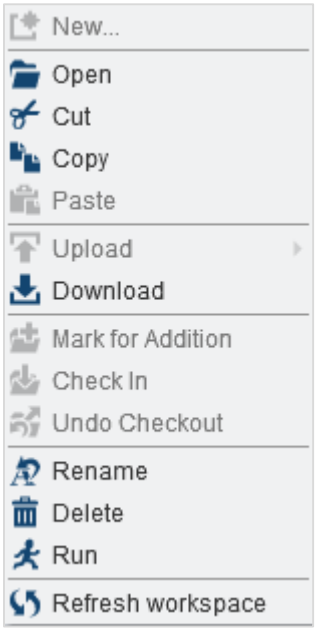

Figure 6. Example Workspace menu for a file

A file also has to be marked for addition prior to copying it to the Repository using the *Check In* option. If the file already exists in the Repository, there is a visual indicator and you are restricted to check the file in. We shall see that this extra, and seemingly unnecessary, step can be quite useful to reserve a file name.

Both the Repository and Workspace menus and the toolbars provide all the necessary features to transfer, synchronise, move and manage content. In the end, the Repository and Workspace is treated as one feature even though they have clearly defined and separate roles in SAS Drug Development.

DIFFERENT APPROACHES

There is no one way to work with the Repository and Workspace. However, a few simple tricks can greatly simplify using either.

It is far easier to create a folder in the Repository first as this will allow empty folders. To create a Repository folder, or folder structure, in the Workspace, use *Get Latest Version*. Manually creating a folder structure in both locations is a tedious exercise and prone to errors.

Get Latest Version also copies all the content of a folder. As there is no option to clone a folder structure, you either transfer it all and then clean (contradicting the previous paragraph) or create a little utility to generate an inventory of the source folder structure in the Repository and re-create it in the Workspace.

Version control is a very useful feature, whether it is used as content is delivered or part of the standard working process. Version control can be very efficient to record milestones or whenever content is added to the Repository. Irrespective of the approach and conventions, avoid unnecessary versions as SAS Drug Development stores the entire file and not the change, or delta. If, for any reason, you have two 1 Gb tab-delimited file with only one digit different, the storage consumed is 2 Gb instead of 1 Gb plus that one digit. A good standard for how version control utilised is therefore prudent.

The *Check Out* option is very valuable when you need to investigate an issue or other fact, especially for a SAS program. Keep in mind that you cannot update a program, or any file, in the Repository from the Workspace until it is in the checked out state. If you have edited a file in the Workspace and then use the *Check Out* option, your changes in the Workspace will be overwritten with the Repository file without warning. A good practice is to check out the file before investigating as you can always use the *Undo Checkout* option, if there is nothing to change.

One feature that is missing in SAS Drug Development is the option to revert a file to a prior version. Until a representable feature is added, one approach to revert a file involves a few manual steps.

1. *Get Specific Version* to obtain the desired version and store it in the Workspace;
2. Make a copy of the file;
3. *Check Out* the current version to allow the Repository file to by updated;
4. Overwrite the checked out file with the copy made previously in step 2;
5. *Check In* the updated file;
6. Remove the file created in step 2.

Reverting a file to a prior version is a frequent task with the above steps quickly becoming second nature.

As the Workspace is in use constantly, content that is no longer necessary will surely accumulate at a very rapid pace and persist until it is intentionally deleted. A large Workspace can be difficult to manage and coordinate with the Repository. The Workspace is much more useful if it is kept clean, only retaining the content absolutely necessary for the current activity. Any other content is only a menu option distant, so retrieving additional data sets, programs, etc. is quick with little effort.

Many organizations consider a general rule that all delivery and sharing of content is performed through the production area or Repository. As deliverables are completed and published to the Repository, ensure that the corresponding content is removed from your personal Workspace. When all activity is complete, the Workspace for the project, study or analysis should be empty.

PRIVILEGE, MEMBERSHIP AND PERMISSION

Any analysis environment requires some form of access control and governance and SAS Drug Development as an application is not different. The application implements a scheme that manages access to the application features and content through three separate mechanisms; the privilege, member and permission. We shall only briefly discuss the approaches and conventions as any detail will be very specific to the requirements of each organization.

PRIVILEGES

A privilege, in general terms, is granting access to a SAS Drug Development feature. For example, if you are granted the privilege *Create SAS Session*, you are allowed to start a SAS session, execute a SAS program or, interestingly, use the Data Viewer to view a SAS data set. The latter is backed by a SAS session.

A privilege has two scopes, or contexts, in that privilege can be granted for global application features or specifically for features how they apply to the content within the Organization, Project and Analysis containers.

A container privilege is not granted to a user directly as is a global privilege. A container privilege is associated with a Role within the Organization or a specific Project or Analysis, to which a user is a member, either directly or through a user group, and thus granted the privilege. The only restriction is that a user must be a member of the container. This greatly simplifies the process to assign consistent privileges to users performing the same business role or function.

A privilege for the Organization and Project is inherited to the next container, either automatically or by having the container inheriting the Role. Careful consideration and a testing is necessary when setting up roles to avoid escalated privilege.

It is worth to mention that the use of Roles is solely applicable to privileges. A Role is currently not used anywhere else in SAS Drug Development.

MEMBERSHIP

A membership is applicable to containers, e.g. Organization, Project and Analysis, and serves two principal purposes. If we consider users as a *member* of an organizational group or project and analysis teams, the concept of membership is an efficient mechanism. Users that are part of a team should be provided or be eligible to access content, be assigned tasks and contribute.

SAS Drug Development concurs, thus a member of an Organization or specific Project and Analysis containers can access content, be assigned tasks and contribute.

A user by default cannot access content in the Repository beyond the user's home folder. To be provided access to content, a user is added as a member to the respective container. All users must be members of the Organization container to be eligible, e.g. selectable, for a specific Project and Analysis container. Similar, a user needs to be a member of the parent Project container to be granted access to that projects Analysis containers. Membership is not inherited by default.

As a member of an Organization, Project and Analysis container, a user is also granted member access (see discussion on permission below) to content within the container, hence ability to access the Organization, Project and Analysis content. The user at the same time also becomes eligible to be assigned roles and activities within the container.

ABOUT GROUPS

SAS Drug Development also implements the concept of a user group, which is defined within a container.

A group is simply a named collection of groups and users, which can further be used when granting container privileges, assigning tasks to a group, etc. Support for nested groups is something that may be extremely useful for organizations that have large business organizations or relationships with many third parties.

The concept of global user groups do not exist in SAS Drug Development, except that groups are inherited, e.g. a group defined within the Organization container can be inherited to Project and Analysis containers, if the group is added as a member to respective containers. Similarly, a Project container group is inherited to a project Analysis container, if the group is a member of that Analysis container. Keep in mind that inheriting a group through container membership implies all the members of the user group is granted member rights, which is not always the desired effect.

PERMISSION

Permission in SAS Drug Development is specific to content within the Repository and not solely the content within a container.

Permission (Figure 7) can be defined on each Repository item for the Owner, Members or specified users and groups that are appended to the list. The first two, Owner and Member, are default principals and cannot be removed with the Member principal representing those groups and users that have been specified as members to the parent container.

Permission follows five simple common categories.

- *Admin*
- *Read*
- *Properties Write*
- *Content Write*
- *Delete*

For each, the permission can be granted (Allow), denied (Deny) or inherit, the latter recognised by neither Allow nor Deny selected.

Specific to SAS Drug Development, the owner of a folder or file is automatically granted irrevocable administrative (Admin) permission. As the ownership of a container, folder or file is re-assigned, so is the administrative grant. Depending on how you configure default permission, the owner of a Repository folder or file most likely be the regular user who initially created it.

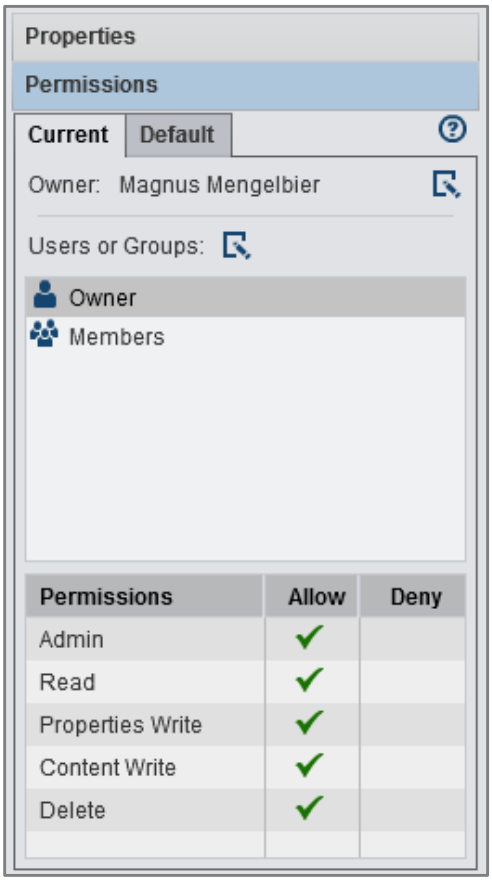

Figure 7. Permission (Folders)

The distinction between *Properties Write* and *Content Write* is quite straight forward. In reverse order, *Content Write* refers to a specific folder or file. *Properties Write* refers to metadata associated with the specific Repository item that includes, but not restricted to, custom attributes. For example, a user cannot or get curious messages when attempting to lock or check out a file, if the user has been denied the permission *Properties Write* since both the lock and check out options are metadata flags.

SAS Drug Development also defines a Default permission for new items added to containers and folders. The default permission is assigned to each item as it is added to the container or folder. The default permission associated with a container, for example Organization and Project containers, are only applicable to a new container at the next level and not applicable to the content within the container. As such, default permission defined on the Organization container is applicable to new Project containers and not the folders and files within the Organization.

The current implementation of default permission can also not distinguish between folders and files, such that default permission are assigned equally to a subfolder and file as it is added.

The specification for default permission uses the same permission categories as above. It is also possible to assign a specific user as the default owner for any new folder or file added to the container or folder. This little feature is extremely useful and welcome as we shall soon discover.

SAS Drug Development uses a pessimistic approach to determine the default permission that is applicable to each new item in the Repository.

- Default permission is no access.
- Read-only access is granted to members of a container.
- Any other permission is granted only if explicitly provided.

If two permission definitions are contradictory, for example no access and read/write access, SAS Drug Development will continue to be pessimistic and opt for no access. Be sure to verify the inherited permissions as well, since they may be the source of confusion as to why one or more users do not seem to have the correct access.

LOCKING OUT THE ADMINISTRATOR

SAS Drug Development, as previously mentioned, grants the owner of a folder or file automatic and irrevocable administrative (Admin) permission. This is a basic system feature and cannot be configured in SAS Drug Development 4.4 or earlier.

It is possible for a regular user that owns a Repository folder or file to remove access, whether intentional or unintentional, for all users, including administrators, managers, etc. When access is removed for a folder, this also applies to all subfolders and files within that folder.

The user interface, SAS Drug Development macros and Java API only display or return those items that a user is granted to access, which is correct and desired behaviour but does further complicate the issue. In our case, the only indication of a folder's or file's existence would be to analyse the verbose audit trail for folders and files that have been created, not deleted and not accessible in the Repository by the appropriate groups or individual users.

The end result is that if access is removed for a sufficiently large group of users, the applicable folders or files are essentially hidden from other users and groups requiring access, auditors and inspections. If the case is critical enough, quality assurance arguments may equate sufficiently revoking access with a soft delete, but that should be determined by each organization independently.

This may not seem like a serious issue unless it is on a folder that contains or a file that represents data sets, programs, outputs or controlled documents that fall within GxP, regulatory or legal stipulated retention and electronic source record requirements.

The scenario can be avoided entirely with implementing a functional user account as the owner of all content in the Repository. As mentioned when considering SAS Drug Development permission, a container and folder can have a user specified as the default owner, which would be our Repository Owner, for all new folders and files. If we now happen to remove administrator access, we can restore it by signing in as the Repository Owner functional user account and apply the appropriate permission. To our benefit, the actions to restore permission are recorded in the audit history for the folders and files affected, so complete traceability is preserved.

An additional benefit of the Repository Owner approach, for those organisations migrating from SAS Drug Development 3.5 or if you already have content in SAS Drug Development 4, is that the single Repository Owner can be applied using a Request For Change (RFC).

GOOD PRACTICES

There are several good practices with respect to privilege and permission that can be adopted or adapted to each organizations requirements.

A general convention in SAS Drug Development is that there does not exist a privilege, membership or permission that corresponds to the super Administrator. Administrator privileges and permission is assigned a user on an individual basis by feature or Repository item. However, it is good practice to create the Administrator role and groups as appropriate.

The steps to apply permission in SAS Drug Development is replace and not append, as we most often are accustomed to. This includes the user interface and specifically the SAS Drug Development macros and Java API. Use the "replace" perspective when defining and applying permissions as it will save a great deal of extra work.

It is advisable to implement the single Repository Owner and a default permission scheme as early in the use of SAS Drug Development as possible. Applying the approach becomes more comprehensive as content is accumulated in the Repository.

The ability to define default permission for new subcontainers, subfolders and files in a container or folder is one of the most useful features. This implies that you can readily define a permission template for new content with a low level of maintenance.

A default permission scheme for the Repository, defined on the Organization container and main folders, can be as simple as

- Default owner to Repository Owner or an equivalent account
- Default Member permission is grant Read (Allow) or deny Read (Deny)
- Default group ADMIN or Administrators with Full permission (Admin, Read, Properties Write, Content Write and Delete)

There are also performance differences depending on how you apply permission that become more discernable as the number of items in the Repository increases. Experience has shown that inheriting permission is more

efficient than explicitly defining permission for each folder and file. As such, design your permission scheme to inherit as much as possible.

Inheritance, or the use of, also makes sense as modifying permission is only applied on a single or limited number of folders and files, instead of each specific Repository folder or file.

THE SAS PROGRAM

One of the main reason for selecting SAS Drug Development as an analysis environment is very likely to be the ability for executing a SAS program and storing any eventual output. As we shall discover, the two main actors surrounding SAS programs are the Workspace and the SAS server. The Repository remains essentially a central location to store inputs to and outputs from the SAS program.

We shall first look at the different SAS Drug Development specifics, concepts and conventions with regards to SAS.

SAS IN SAS DRUG DEVELOPMENT

All SAS processing occurs on the SAS Drug Development SAS servers, so any migration of content follows the same process and subjected to the same issues if you are migrating from a Windows SAS server environment to one based on Red Hat Linux. The specific details is defined for each SAS Drug Development instance as deployment is planned and the compliance documentation is prepared.

SAS in SAS Drug Development 4.4 and later defaults to SAS 9.4 64-bit on Red Hat Linux on Intel based hardware. This should not pose any roadblocks, but consider which and how Microsoft Office file formats, such Microsoft Excel, are used within the organization. Depending on your current process, the change can be minimal or impact can be significant. Good references are those discussing the SAS on Red Hat Linux, as that is where the processing occurs given that the SAS Drug Development application treats the Microsoft Office documents as simple files.

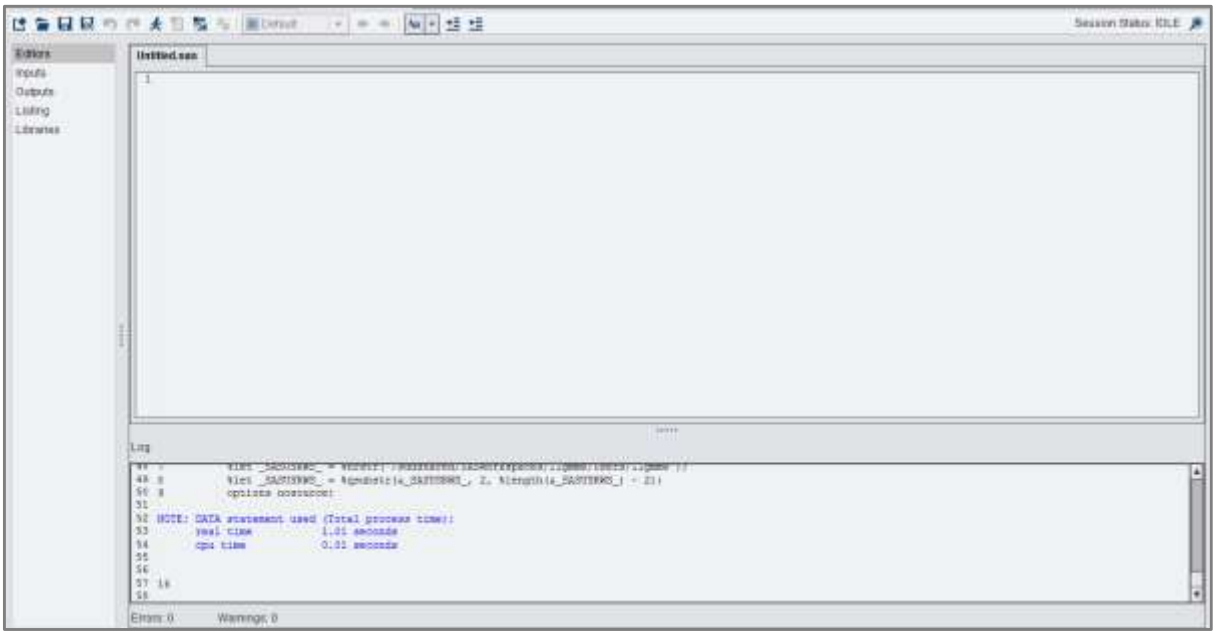

Figure 8. The SAS session

SAS SESSION

The SAS session is pivotal to and the principal service behind all activity in SAS Drug Development that revolve around SAS data sets and programs, regardless if it is interactive or executing as a one-off or scheduled background process.

A SAS session is dedicated to a specific user with a user allowed to have multiple concurrent SAS sessions active at any one time. Each SAS session is also independent of other SAS sessions, which to a regular user primarily means that each SAS session has its own WORK library. SAS Drug Development does not include any

default server resource quotas beyond that defined in the SAS configuration file and, as such, memory, temporary disk space, etc. is shared between SAS sessions on the same SAS server.

The user interface for an interactive SAS session (Figure 8) is also no different than a SAS session on your local PC or a SAS server. You are still provided access to the usual windows such as the log and output windows, libraries, etc.

The integrated program editor does provide some of the expected features, such as multiple program windows, able to execute the entire program or selected code blocks, view library content and data sets. The editor also provides reasonable syntax highlighting that is improving with each version, but should not be expected to be fully comparable to the Windows Enhanced Editor.

One curious aspect is the lack of configurable keys, if you are used to F3, F5, F6, F7, etc. Key are not configurable, but some standard keys are available, even if their function depends on which Web browser is in use.

SAS Drug Development does have its nuances, requirements and dependencies. The first noticeable difference is three global macro variables defined when the SAS session starts.

```
54          %put &_SDDUSR_ ;
someuser

55          %put &_sasws_ ;
/sddshared/SASWorkspaces/someuser

56          %put &_sasusrws_ ;
/sddshared/SASWorkspaces/someuser/Users/someuser
```

The macro variable _SDDUSR_ contains the SAS Drug Development user name that is associated with the SAS session, e.g. the SAS Drug Development user running SAS.

The macro variable _SASWS_ contains the root folder for your personal Workspace, or the workspace in which SAS executes. We shall look at this in greater detail shortly.

The variable _SASUSRWS_ contains the path for your home folder in your personal Workspace, or the workspace in which SAS executes. Just because this value is defined, does not necessarily mean the path exists in the Workspace.

The user name in _SDDUSR_, _SASWS_ and _SASUSRWS_ is not guaranteed to be the same as SYSUSERID. SYSUSERID is an operating system compatible user name as the actual SAS Drug Development user name may include characters that are invalid to the operating system. The user name stored with SYSUSERID is still unique, but you should consider the more correct _SDDUSR_ for programming purposes.

Another restriction is for the operating system access and its commands. They are not accessible, so no possibility to use systems commands to, for example, obtain a listing of files in a specific folder or to copy, move or rename files.

All else should be as any other SAS environment, including paths. If you are accustomed to SAS Drug Development 3.x, the cryptic paths have disappeared and the tricks with the SDDPARMS data set are no longer necessary to make any sense of folder structures and paths.

THE JOB

The Job is a SAS Drug Development feature to execute a sequence of SAS programs and is, more or less, equivalent to one program calling multiple %INCLUDE statements to execute a set of programs in a given sequence.

The Job is provided with a user interface (Figure 9) as it serves a dual purpose that is slightly more complex than just the set of %INCLUDE statements. It is the only method to execute a SAS program as a batch or background process. It is also the only mechanism available to execute a SAS program directly in the Repository, which we will discuss in more detail further on.

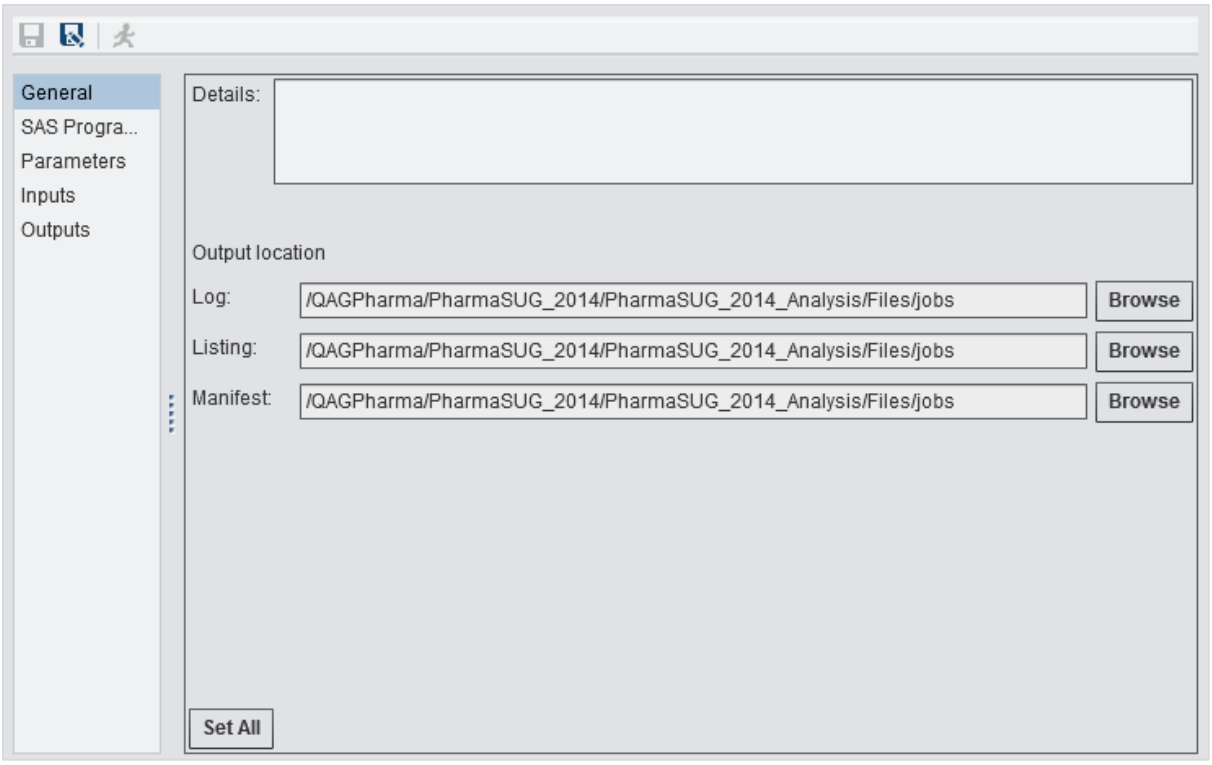

Figure 9. The Job interface

One important aspect is that each Job is provided a separate unique SAS session and all SAS programs in a Job executes within that single SAS session. SAS Drug Development does not reset or perform any cleaning, for example removing WORK library data sets, between SAS programs in a Job, which can be a source for confusion as to why the result is not correct or the program suddenly stopped working. On the other hand, it also provides a high degree of flexibility in how you set up SAS programs, if managed properly.

As a Job executes within a single SAS session, the Job will produce a single log, listing and manifest, which output location is specified through the Job general settings.

The manifest is a SAS Drug Development specific output that details all the inputs, programs and outputs for the Job as it executed. It can easily be reviewed in the SAS Drug Development interface, or if you are so inclined, both the Job and the manifest are eXtensible Markup Language (XML) files that can be programmatically analysed.

The Job editor interface makes it easy to define a Job that executes a single or sequence of SAS programs (Figure 10). When specifying a SAS program, there is notably no restriction that the program location has to be in a specific relation to the Job.

The SAS program location can be a path specified as relative to the Job location ('./') or absolute. A relative path can be useful if creating template Jobs that rely on and can be copied into a standardised directory structure. As a Job executes, any and all relative paths are translated into absolute paths, which is the record in logs and manifests.

A Job may also contain parameter definitions. These are exposed to the SAS session and your SAS

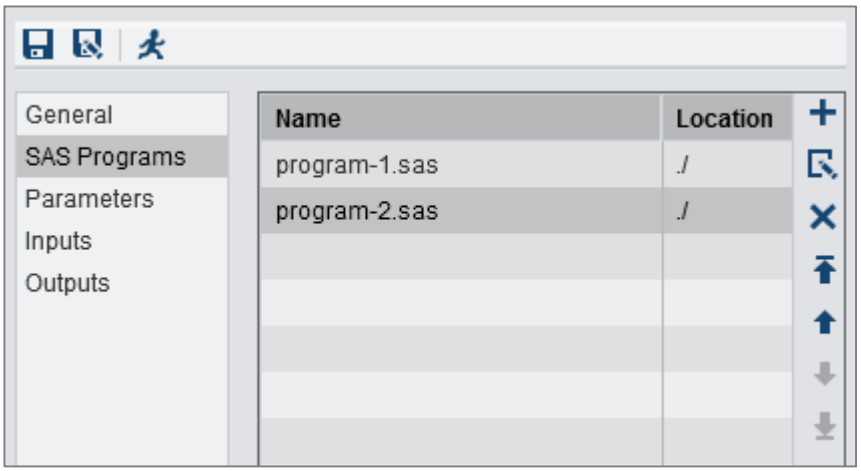

Figure 10. SAS programs in a Job

programs as global macro variables. This implies that your parameter names need to be valid macro variable names. The parameter value is also passed to the SAS session as is without quoting, so any special characters need to be macro quoted appropriately using the regular macro quoting functions. A warning or an error in the log prior to the first line of your program or the first lines of code not executing is usually a good indication that your macro parameter quoting contains a syntax error.

General settings, SAS Programs and the optional Parameters are the only configuration items that are required to execute a Job within the personal Workspace. The additional configuration options Inputs and Outputs are required and used when executing a Job directly from the Repository.

THE JOB AND THE REPOSITORY

The Job, as we previously hinted, can be employed to execute one or a sequence of SAS programs directly from the Repository without resorting to the personal Workspace. Before we consider the particular options, let us first look at how SAS Drug Development executes a Job from the Repository.

As it turns out, the personal Workspace is not the only workspace. There also exists a transient workspace that is created every time a Job executes from the Repository, as seen in the extracted SAS log below (emphasis added).

```
36         %put &_SASWS_ ;
/sddshared/SASWorkspaces/.transient/transient_workspace.job-75794630-2365-4e30-b510-
2686d9276de1
```

The transient workspace is a simple approach to ensure that the SAS programs within a Job are provided all the correct inputs and output locations and will not impact other activities.

The transient workspace is by default empty and, as you recall, workspaces are not linked to the Repository. SAS Drug Development cannot predictably guess what input files your program requires or guess what outputs you create are temporary or should be retained, hence the Job configuration settings for Inputs and Outputs.

The Inputs configuration of a Job specifies a folder or file to be placed in the transient workspace prior to the first SAS program executing. If you require, you may specify that all subfolders should be included as well. However, it may not be fruitful to select the entire analysis folder, with *Include subfolders* selected, as an Input location since everything will be copied to the transient workspace each time the Job executes.

The location in the transient workspace will use the same path as in the Repository except that the path starts in the folder specified by the _SASWS_ global macro variable.

For example, if you wish to assign a SAS library to the data set folder

```
/QAGPharma/PharmaSUG_2014/PharmaSUG_2014_Analysis/Files/final/data/adam
```

then specify the LIBNAME statement

```
libname mylib
"&_sasws_./QAGPharma/PharmaSUG_2014/PharmaSUG_2014_Analysis/Files/final/data/adam" ;
```

The macro variable _SASWS_ will refer to the personal Workspace whenever the SAS session is an interactive session or associated with a Job that is executing as a background process from within the personal Workspace. Otherwise, it refers to the transient workspace created specifically for a particular Job executing from the Repository.

The Outputs configuration option, including the *Include subfolders* option, follows the same convention as Inputs except that you can only specify a folder. Interestingly, the target folder does not have to exist in the Repository when the Job is edited in the Workspace. If there is a file to return, the missing portion of the folder path will be created.

After the last SAS program completes execution, SAS Drug Development will attempt to identify new or modified files to return to the Repository. SAS Drug Development cannot (as of version 4.4) return an empty folder created while executing the Job.

SAS Drug Development will also not distinguish between Input and Output locations. If a file is modified and contained within an Input location and is not specified as part of an Output location, the modified input file will still be updated in the Repository.

For example, if you sort an input data set, that data set will be updated in the Repository regardless if the library folder is specified as an Output location. To circumvent this, simply ensure that all input locations are accessed read-only, such as the using the ACCESS = readonly option in a LIBNAME statement.

WORKING WITH SAS PROGRAMS

The approach to creating, modifying and executing SAS programs is as much a corporate culture and standard as personal. As you will undoubtedly execute programs from with the Repository, a SAS Drug Development Job and associated SAS programs are indistinguishable.

The general approach to analysis programs (Figure 2) can also be applied to a SAS Drug Development Job (Figure 11).

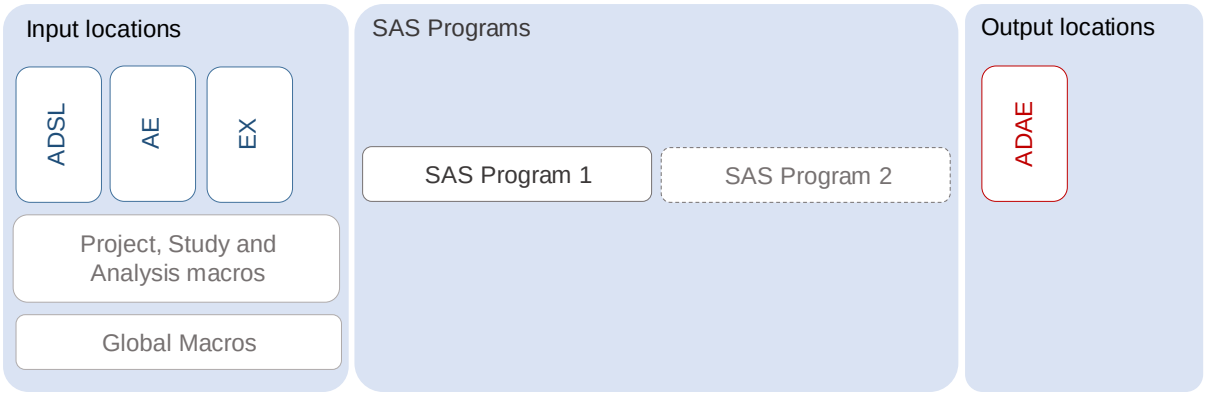

Figure 11. SAS Drug Development Job

Our inputs would include all the data sets, input files and any macros we will require. The latter is a requirement since SAS Drug Development does not contain a standards library feature.

Plan to keep the Job simple as this will also simplify setup and maintenance. There is currently no restriction to the number of Jobs in SAS Drug Development or that a SAS program can only be a member of a single Job. A reasonable approach could be to use a single Job for each program under development, one Job to execute all programs for a specific deliverable, such as SDTMs, ADaMs, Tables, Listings and Figures, and one for the entire analysis.

As you create or start editing a SAS program, add it to the Job along with any input and output locations. If you are reading in a specific file, use the file rather than the entire folder as an input location. If the file is not there in the future, the Job manifest and log messages will report the missing input.

When complete, publish (*Check In*) the Job and SAS programs to the Repository and execute the Job. This should ensure that your Inputs and Outputs are correctly defined early in the process.

A very easy pitfall is to copy a Job and not review the Job definition and configuration. All configurations and definitions are retained, which may mean that output is written to an entirely different location than expected. A good practice is therefore to use relative paths as much as possible.

An extremely bad practice is to hard code paths in programs to access a personal Workspace when executing a Job in the Repository. The Workspace is only accessible by a single user, so another user will not be able to execute the program. In addition, Workspace content is not registered in the audit trail, so there is zero traceability. The best approach is to ensure that a SAS program only accesses content from where it is executing, either the Repository as part of a Job or in the Workspace.

One restriction worth noting is that you cannot use the SAS Code Analyzer procedure (PROC SCAPROC). Doing so will result in a SAS Drug Development functionality failing with curious behaviour. SCAPROC is the magic behind how the SAS Drug Development Job identifies what files have been modified. The root cause is not with SAS Drug Development, but in that the SCAPROC procedure cannot write to multiple destinations.

STANDARD LIBRARIES IN SAS DRUG DEVELOPMENT

The lack of a standards library feature is a tangible difference to users as any macro or corresponding folder will need to be listed as an input location. If a macro library is fairly simple, the impact may be negligible. As the complexity of the macro library increases, the impact will follow suit as it may not be efficient to include large

folders or feasible to include individual macro references as input files, especially if you have several levels of nesting and/or the macros are highly parameterised and data driven.

Many, if not the majority of, SAS environments rely on the SAS Autocall macro facility, or similarly macro catalogs, to ensure the correct macro library is available to an executing program with little or no input from the user. The configuration may also be conveniently hidden in a setup or initialization macro or program, further liberating the user.

It is possible to require that each user stores a personal copy of the entire macro folder or catalog in their Workspace. Any synchronisation routine is managed and executed by the user to which the Workspace belongs as the Workspace is not reachable by and outside the control of the central administrator. This approach would also require hard coding the macro folder or catalog path since transient workspaces are empty by default.

The synchronised library copy is therefore not ideal and prone to possible compliance issues since it is multiple uncontrolled copies of macros beyond administrator's control and possibly at different states of completeness, which is probably the very reason the central standard library was created in the first place.

The Central Library Manager (CLM) is a simple approach to implement a central library in SAS Drug Development 4. A special folder is created in parallel to both the personal Workspaces for users and the location of the transient workspaces. This is usually a single location, which means a single workspace library folder to maintain.

The workspace library folder is readable by all users except a single functional account, the Library Manager, who has permission to create, read, modify and delete the content of the workspace library folder.

The Library Manager account owns a scheduled Job that will replicate, or mirror, the entire content of one or more source folders in the Repository. The process can be optimised such that the replication schedule may be every 10 minutes, if so required.

The only drawback with the CLM is that SAS Drug Development currently does not provide a user interface to browse the workspace library folder. It is however, extremely simple to generate a CLM report of which files are currently stored within the workspaces library folder(s).

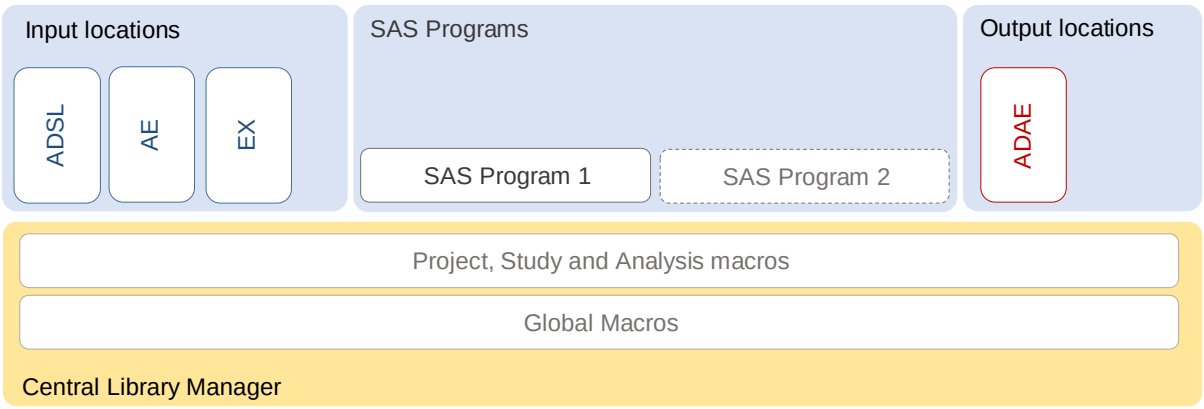

Figure 12. SAS Drug Development Job with the Central Library Manager

The SAS Drug Development Job combined with the CLM (Figure 12) is one approach to implement the general approach to analysis programs (Figure 2) discussed in the beginning of the paper. The CLM also provides the seamless access to standard libraries, of which macros may only be one of several components that users have come to expect from SAS analysis environments within the Life Sciences.

CONCLUSION

There is no singular recipe to evaluate or use SAS Drug Development in an organisation. It is a comprehensive solution with many more features than discussed herein and what is within scope or achievable will vary greatly between organisations. This is no different than analysis environments based on SAS server and/or SAS on a workstation.

The preference for SAS Drug Development is most often a business decision, although a key factor is often its capability to execute a SAS program and, at the same time, store both the program and any inputs, outputs and other associated documents in an integrated repository. The approach to trace and address the use of SAS programs, taking into account business processes, conventions, standards, libraries and programming culture, in

general, has highlighted the solution benefits and practices that can be employed or those that need to be managed in order to simplify use.

The result is a well formed view on SAS programming requirements in addition to a thorough evaluation of regulatory compliance requirements, directory structures, business roles, privileges, permission and overall governance. This can also highlight those SAS Drug Development features that are identified as a requirement, a nice-to-have, the not so useful, and those missing but desirable.

An important consideration is that a business process can evolve within SAS Drug Development as well as both adopt and adapt features as the process matures. The SAS Drug Development solution remains foremost a repository with the ability to execute SAS programs, which is consistent with and the requirements for our initial business process conventions and outputs.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Magnus Mengelbier
Limelogic AB
Jungmansgatan 12
211 19 Malmö
Sweden

E-mail. mmr@limelogic.com
Tel. +46 40 685 1480
Web. www.limelogic.com