Paper 178-2012

Sharing SAS® Programs Between PC, Server and SAS Drug Development

Magnus Mengelbier, Limelogic Limited, London, United Kingdom

ABSTRACT

SAS Drug Development has become a common SAS environment within Life Science organisations, whether large or small. It is not uncommon that programs are developed outside of SAS Drug Development, either as a standard process or simply that an external organisation is not allowed access to the SAS Drug Development environment.

The convention to configure parameters for any SAS program to be executed within SAS Drug Development provides a logistical exercise in retaining context for projects, compounds, protocols, reporting event, etc. We discuss a convention and consider example code that allows for the context to be retained for a SAS program, whether it resides within SAS Drug Development or executed on a SAS PC or server.

INTRODUCTION

A statistical computing environment (SCE) based on SAS can come in many different shapes and sizes, either as SAS on a Microsoft Windows workstation, a server based on Unix, Linux or Microsoft Windows Server or enterprise solutions such as SAS Drug Development.

SAS on a workstation and a server share a common trait in that the storage of data sets, format catalogs and programs are most commonly located on a local or network drive or share. SAS Drug Development contains a comprehensive repository that adds functions such as audit trail, version control and other compliance functions.

Common to all environments is some form of configuration requirement and often implemented by a standard utility in order to initialise or define data, format and macro libraries, standard output destinations and other configurable options specific to your process.

SAS on a workstation and a server most often rely on an autoexec, standard project specific include files or one or more standard initialisation macros in order to define the appropriate environment for a specific program. SAS Drug Development uses configuration parameters and variations of standard and program specific code, which together perform the duty of configuring the program environment.

SAS DRUG DEVELOPMENT

SAS Drug Development is a comprehensive system that consists of several moving parts (Figure 1). Each moving part, usually referred to as tiers, contributes to the whole system in a well-orchestrated and integrated way. The three main parts that will explain the role of configuration parameters are the repository, Web-tier and the SAS execution tier.

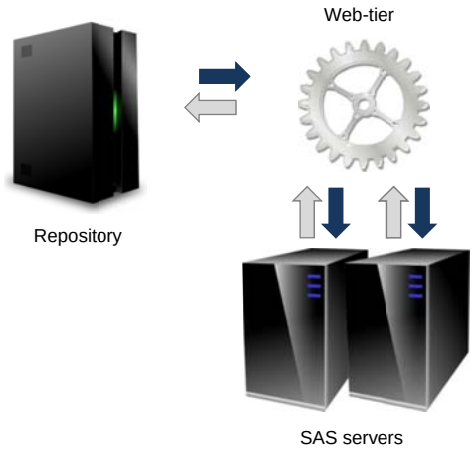


Figure 1. SAS Drug Development main components

SAS DRUG DEVELOPMENT PARTS

The repository contains all content, except those tools that you decide to deploy on special location on the SAS server or servers. Two or more SAS servers, the SAS execution tier, retains the sole responsibility of executing your SAS programs and supporting the SAS Drug Development tools such as Process Editor, Data Explorer, parts of the Advanced Loader, etc.

Interestingly, the SAS servers do not by default have direct access to the repository, e.g. the location of data sets, programs, etc., in SAS Drug Development[1], which differs from how regular SAS PC and server environments are configured. Any inputs and outputs have to consequently be temporarily copied from the repository prior to the SAS server executing your program (dark arrows) and then any outputs moved into the repository after execution ends successfully (grey arrows).

[1] The disconnect between the repository and SAS servers is until and including at least SAS Drug Development version 3.5.

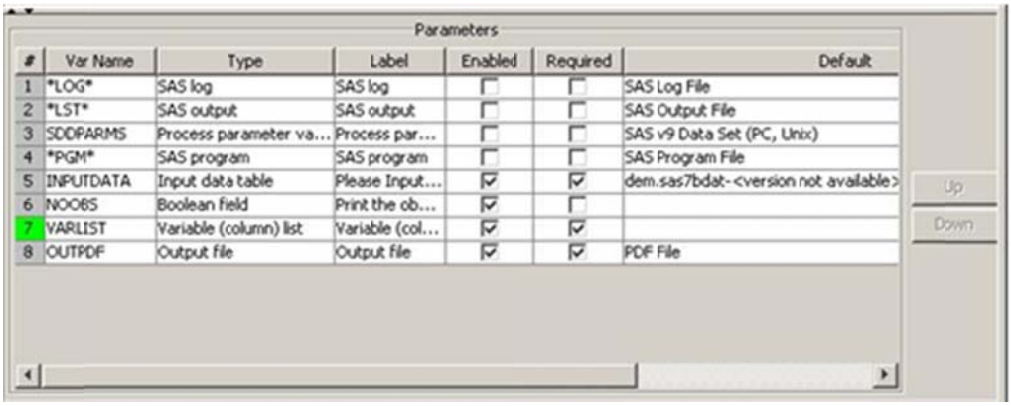
A direct connection to the repository can be established with care from within a program, both from within SAS Drug Development and externally from SAS on a workstation or server, but this will introduce issues with traceability and auditing as the connection is not recorded as part of the audit trail for the program execution.

Another perspective, which will become important to consider further on, is that the user of SAS Drug Development will next to never connect directly to the SAS server and perform all communication and activities through the Web-tier.

Rather than to inspect the SAS code to determine inputs and outputs, SAS Drug Development relies on the user to specify all inputs required and outputs to be retained and this is the simple role of configuration parameters.

CONFIGURATION PARAMETERS

Configuration parameters in SAS Drug Development (Display 1) define all aspects surrounding the program, which include all inputs, program "configuration" options and all outputs that need to be returned and saved to the repository. If you omit to specify an output, the output will not be moved to the repository after the program ends executing.



Display 1. SAS Drug Development configuration parameters

A configuration parameter has several attributes that are specific to what input or output item it represents, but common to all is the name. A program can reference a configuration parameter in two different ways. A global macro variable with the configuration parameter name is automatically created, which value points to the temporary location on the SAS server.

The configuration parameters LOG, LST and PGM may not be accessible through global macro variables, but can be obtained in the SAS Drug Development parameter data set as well as the original value of any configuration parameter. The special parameter data set is created for each execution of the full or just a selection of program code and contains all the original parameter values and other house-keeping properties. The data set is of course accessed by specifying the global macro variable SDDPARMS instead of the data set name.

The configuration parameters are actually stored within and not separate to the program file in the repository as an eXtensible Markup Language (XML) structure appended to the end of each SAS program. The XML structure, in earlier versions of SAS Drug Development referred to as PACMAN, is interpreted each time a SAS program is executed within SAS Drug Development as the initialisation step. This XML structure remains well hidden from the user and is edited through one of the SAS Drug Development user interfaces, such as the Process Editor or Job Editor, rather than coded by the SAS Drug Development user.

The difference between how programs are configured between the environments SAS Drug Development and SAS on a workstation or server prohibits a SAS program from being uploaded into SAS Drug Development from a local network share and executed without change. A next logical step is a simple convention that would allow a SAS program to work within and outside of SAS Drug Development and let the upload and download process complete any technical transformations necessary.

PROGRAMS FOR BOTH

An efficient process would incorporate initialisation requirements for both SAS on a workstation/server and SAS Drug Development, making the transition between the systems seamless to a program and the programmer while in a very easy way accommodate all the technical nuances of each source and target environment, regardless if you are a sole organisation or rely on the assistance from Contract Research Organisations (CROs).

It could hypothetically be possible to construct an initialisation macro that would read and parse the PACMAN XML structure for use with SAS on a workstation or server, but that is an extensive and not so user friendly exercise. The XML structure and content is very verbose, unnatural and a technical facility for most users. Programs would also then be locked to a specific version of SAS Drug Development as we can only expect the configuration parameters and the XML structure to evolve with new features, customisations and enhancements.

APPROACH

A convention should ensure that the program and the internal references in the program are consistent for all three environments, e.g. workstation, server and SAS Drug Development. The basic principle is that any item that is required to be defined through SAS Drug Development configuration parameters is also defined through macro calls in the SAS program as executed on a workstation or server.

- When the program is imported into SAS Drug Development, the macro calls are interpreted and used to create the PACMAN XML structure.
- When programs are exported out of SAS Drug Development, the PACMAN XML structure is parsed by a standard utility and replaced with macro calls.

Instead of one massive macro or defining the same “configuration” item in several places, a few small macros (Table 1) have been developed that will allow a flexible consistent approach to define all the required configuration parameters whether within SAS Drug Development or SAS on a workstation and server.

Macro	Description
%meta()	Defines context information that is usually derived from a program location within a directory structure
%library()	Defines a SAS library for accessing one or more data sets and format catalogs
%folder()	Defines an input and output folder
%dataset()	Identifies a data set for required input data set, resulting output data set, or both.
%file()	Defines an input file
%output()	Defines an output destination folder or a specific output file.
%define()	Identifies one or more global macro variables as not a configuration parameter.

Table 1. Configuration macros

The %dataset() macro is a tweak as SAS Drug Development configuration parameters can point to an individual data set without specifying a library.

%META() MACRO

A program in itself is most often not aware of the context, such as sponsor, compound, project, indication, protocol, stage, purpose, etc. There are two basic issues with today’s implementations, convention and consistency.

The context can be derived from library name definitions, output destinations, etc., if the directory and folder paths are defined within the program. As is most often the case, conventions dictate that paths, library definitions

and output destinations are all or partly defined in the initialisation step outside of the program. You may also not agree with the example list that define context as example terms may include synonyms or terms you do not use, which terminology and use are not consistent across Life Sciences – there are no *transport files* for a SAS program.

The %meta() macro aims to resolve and define context within each program and relies on the simple principle that a protocol reference should be unique in a way that it can be used to identify other context elements, such as compound, project, indication, sponsor, etc. in order to navigate the company standard directory structure (Figure 2). In addition to the protocol, references to the reporting event, purpose and stage is also captured, with the two latter most often synonymous if it is a development, quality control (QC) and production ready program.

```
%meta( protocol = L0002-E00-016,
       event = ind,
       purpose = qc,
       stage = dev ) ;
```

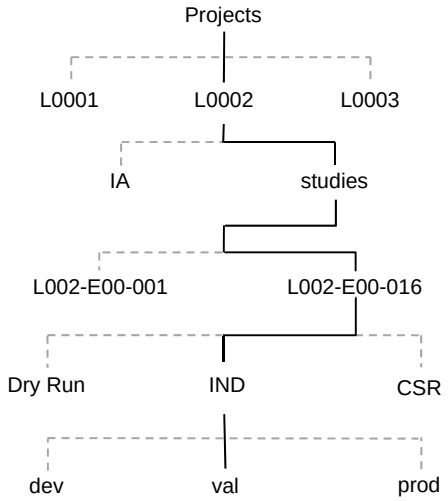


Figure 2. Example directory structure

It is important to note that the %meta() macro defines and uses consistent terminology that is then mapped to the appropriate directory structure, e.g. the purpose of *QC* versus the folder *val*.

Additional optional attributes can be added to the %meta() macro, such as tags if you implement Source Code repositories or other storage that allow for tagging of content.

%LIBRARY() AND %DATASET() MACROS

The %library() macro creates a library reference to a specific path. Technically, a SAS library, or libname, is a shortcut reference to a folder that may contain SAS data sets or catalogs and is consistent across all SAS environments.

The %library() macro will create a SAS library references with the specified library name to the specified folder, if the library does not exist. This allows for libraries to be initialised using standard conventions, such as an autoexec, and the program to retain the library reference when imported into SAS Drug Development.

SAS Drug Development also allows files to be included or excluded to be specified. This is very beneficial if the library folder contains rather large data sets that are not used and transferring prior to execution is unnecessary overheard.

The library path can be constructed using the components defined through the %meta() macro, which supports common concepts in autoexec and other initialisation macros.

The %dataset() macro is, as previously mentioned, a tweak since SAS Drug Development allows you to reference a data set without a library reference.

ADDITIONAL MACROS

There are additional standard macros beyond the two special %meta() and %library() macros that provide simple but important facilities. A common convention to all of the additional macros, which is dictated by the configuration parameters, is that all references to folders, files, etc. are by a global macro variable name.

The %folder() macro creates a reference to an input and output folder. Note the convention with the hash ('#') reference, which point to standard reference created by the %meta() macro. The reference *#root* refers to the reporting event main folder.

```
%folder( name = listings,
         folder = #root/output/listings,
         type = output ) ;
```

The %output() macro creates references to one or more output files in a folder. If more than one file is specified, and only one reference name is specified or the number of references does not match the number of files, a numeric counter is appended on the reference.

The %output() macro example below will create references DM, AE, PK1 and PK2.

```
%output( name = DM AE PK,
         folder = #root/output/tables,
         files = (dm.doc, ae.doc, pk_c1.doc, pk_c2.doc  ) ;
```

The %define() macro is a utility to tell SAS Drug Development, when the program is imported, to ignore a global macro variable as it is not intended to be a configuration parameter. If a global macro variable is not *ignored*, SAS Drug Development will force the user to define it as a configuration parameter.

PROCESS

The process to import and export a SAS program into or out of SAS Drug Development is designed to use existing SAS Drug Development and standard utilities without special software or parallel systems – simply relying on SAS/Base and the capabilities of a SAS Drug Development process.

Each user has a personal *Inbox* that is used as a single consistent transfer point – the transfer bin. From experience and simple experimentation, the transfer bin can further be divided into project specific areas for each user to simplify the logistics of exporting and importing programs as it is common occurrence that a user is active in several simultaneous projects and reporting events.

EXPORT PROCESS

A user uses a standard SAS Drug Development export process (Figure 3) to select one or more SAS program files to export. The standard process will copy the program file(s) to the transfer bin and while doing so use the original program location and the PACMAN definitions to create a set of macro calls that are added to the SAS program just below the header. The original PACMAN code is not retained.

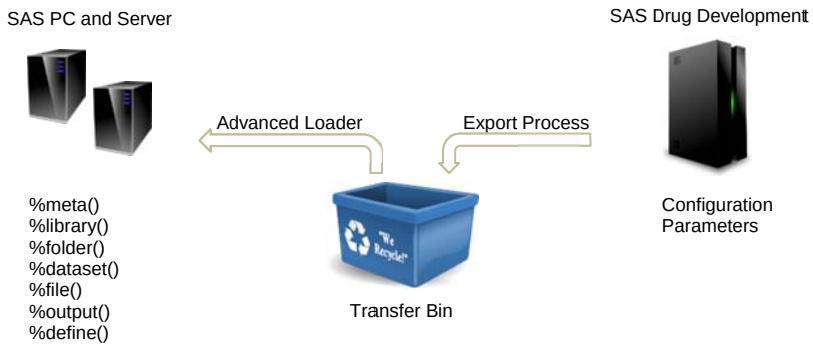


Figure 3. The export process from SAS Drug Development

The user is then able to transfer the file from the transfer bin to the appropriate external destination location outside of SAS Drug Development using the Advanced Loader or other means such as SAS Drug Development Desktop Connection or WebDAV.

IMPORT PROCESS

The import process (Figure 4) is simply the reverse order of the export process. The import process is initiated when a user imports, or loads, a SAS program into the transfer bin using the SAS Drug Development Desktop Connection, WebDAV, Advanced Loader or some other means.

The user then uses a standard SAS Drug Development import process to select the SAS program file to complete the import process.

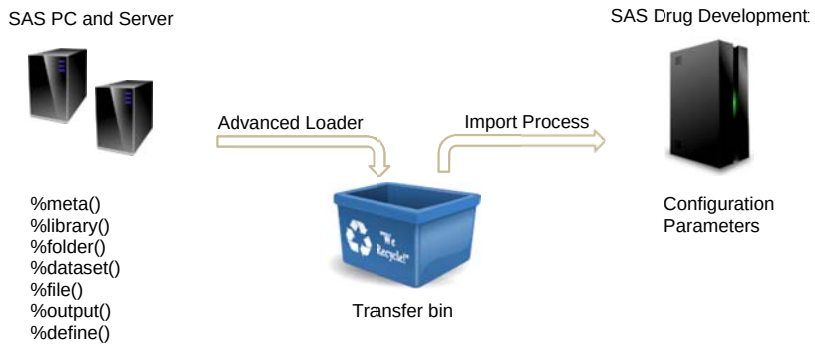


Figure 4. The import process into SAS Drug Development

The standard import process will copy the program file from the transfer bin to the appropriate program location in the SAS Drug Development repository and while doing so use the set of standard macro calls to create and append the PACMAN definitions. Similar to the export process, the original macro calls are not retained.

COMPLIANCE CHECKS

The bi-directional approach allows for very simple import, export and sharing of SAS programs between SAS Drug Development and external SAS environments, whether on a workstation or server or whether sponsor or CRO.

The process also ensures that a program is native to the current location and environment and does not retain any out-dated configurations, if the program is copied or moved to a different project or location. Incorrect configurations can be identified and avoided early in the program development process, if the standard macros are used to setup, configure and initialise the SAS program environment when not in SAS Drug Development.

EXTERNAL SAS DEVELOPMENT ENVIRONMENTS

The approach will also allow for a very simple check-out / check-in process when using external local SAS environments as a development environment and SAS Drug Development as the production instance and code repository since the manual intervention required to move programs between SAS Drug Development and external SAS environments are mostly eliminated.

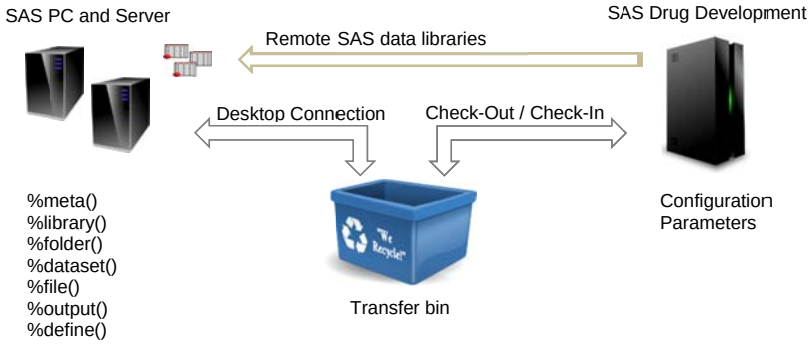


Figure 5. Integrating SAS environments with SAS Drug Development

The %library() macro is extended with the option to *connect* a SAS library to a remote location in the SAS Drug Development repository from the *local* SAS session (Figure 5). The current implementation of the %library() macro allows for two connection options, direct access or cached library.

A direct connection assigns and uses a WebDAV library to access data sets that are located in the SAS Drug Development repository. This approach may suffer from performance issues with rather large data sets as well as program styles that continuously access the data sets in the WebDAV library throughout the program.

The cached library uses a local temporary copy of the selected data sets – recall that SAS Drug Development can specify to include all or a select list of data sets – which only requires the data sets to be transferred once and only when the %library() macro is called. Each subsequent call to the %library() macro can check if a data

set has been updated in the repository and only transfer those that have been updated. This will ensure that the local copy stays as synchronised as possible.

SAS catalogs are not supported since a catalog is currently operating environment dependent, e.g. catalogs created by SAS on Linux and Unix cannot be read from a Windows workstation or server, etc. When catalogs can be accessed across environments, these can be included as well.

A next logical step is to extend the remote connection functionality to other macros where it would be logical to allow input sources located in the SAS Drug Development repository.

The ability to save content in or write to the SAS Drug Development repository from an external SAS environment is currently not supported by default as the entire approach is to allow SAS environments outside of SAS Drug Development to act as development environments. This is simply a business process constraint from the original requirements as content created and intended to be permanently stored in the SAS Drug Development repository has to be created by programs executed within SAS Drug Development environment for compliance reasons to ensure that the complete sequence of activities are recorded and traceable in the audit trail.

CONCLUSION

The convention provides a simple and flexible approach to share SAS programs between SAS Drug Development and external SAS environments while ensuring that programs are native to the SAS environment they currently reside in. Extending the approach to implement local development environments and a central production environment or repository based on SAS Drug Development is also possible with minimal effort. The approach would also greatly improve and promote open project standards and global collaboration as a CRO would be able to use their own standards, SAS environments and publish SAS programs to the Sponsor SAS Drug Development instance without requiring significant investment in tools, utilities and resources dedicated to the Sponsor SAS Drug Development platform.

CONTACT INFORMATION

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

Magnus Mengelbier
Limelogic Limited
Regent House, 316 Beulah Hill
London SE19 3HF, United Kingdom

Phone: +44 208 144 5701
E-mail: mmr@limelogic.com
Web: www.limelogic.com