PhUSE 2015

Paper TS01

Metadata Snapshots: How SAS Drug Development API Macros Can Help

Melissa R. Martinez, SAS Institute, Cary, NC USA

ABSTRACT

During the programming lifecycle of a clinical trial, data and programs may go through several iterations, often marked by specific snapshots or time points in the trial. Sometimes regulatory reports are submitted at interim time points, and it is important to be able to reproduce the programming output. But data and programs often change during that trial lifecycle; metadata may change, programs may need to be modified to deal with changing data. How do you keep track of the data and programs used for a snapshot? Often, an additional copy of the data and programs are stored in a separate folder. This paper offers another alternative: using SAS Drug Development API SAS Macros to store metadata about file versions used for the snapshot, which can be used to easily retrieve those versions later, eliminating the need to create an additional copy of data and programs.

INTRODUCTION

During the programming lifecycle of a clinical trial, data and programs may go through several iterations, often marked by specific snapshots or time points in the trial. Programs are developed using early cuts of data, but sometimes they need to be revised as new data arrives. Many different scenarios could result in changes that require the programs developed with earlier samples of data to be revised. Sometimes, metadata in the source data changes—for instance, new variables are added or the study enters a new phase in which additional data is collected. Sometimes the values in the data are not what was anticipated during initial development. Sometimes new tables, listing or figures (TLFs), or changes to existing TLFs are requested.

In addition to data changes that require program revisions, there are other situations in which a specific snapshot may be needed. Programs go through a cycle of development, validation, and quality control testing of the output, with the cycle being repeated as many times as necessary during the trial. Trials often have interim reporting required at specific times during a study and reports must often be sent to regulatory agencies. It is important to keep track of the specific data snapshot and version of the programs used to create reporting output, so that the results can be recreated at any time if necessary, even if the program or the data has changed since the report was created.

SAS Drug Development is a clinical data repository that provides a seamlessly integrated SAS programming environment. It offers a variety of features that facilitate the statistical programming of clinical studies, such as file versioning, a Project and Analysis-oriented hierarchy, easily-managed permissions and privileges, and a set of API (Application Programming Interface) SAS Macros. While many of the commonly-used strategies for managing snapshots are still available in SAS Drug Development, this paper explores some of the features unique to SAS Drug Development that offer a more efficient solution to snapshot management. This paper was written using SAS Drug Development version 4.5.1. All macros referenced are available in SAS API Macros release 1.4 and higher, and are also compatible with SAS Drug Development 4.4.

THE OLD WAY

The most common way of storing a snapshot involves making some kind of copy of the data and programs and storing them in an additional location. Storing a copy of the data can take many different forms, although the general concept is the same. Some examples include:

- Make a copy of the entire directory structure for the study and name the copied folder with a descriptive name including the date and purpose of the snapshot.
- Have a “delivery” directory that contains dated subfolders. Copy output (including data) to be delivered to the dated folder. In addition, maintain dated folders for cuts of source data.
- Archive the entire study by creating a read-only zip file of the study hierarchy and storing it in a read-only location. Allow only the IT department write access to the zip file’s storage location, and allow only one user permission to unzip the archived zip file.
- Make an export package using file parsing techniques that gather all of the setup files, macros, programs, and data used for the snapshot, places it into a self-contained hierarchy, and provides instructions for its use.
- Sometimes third-party version control software is used to version SAS programs and/or data. Some of these software programs provide users the ability to capture a snapshot of file versions at any given time.

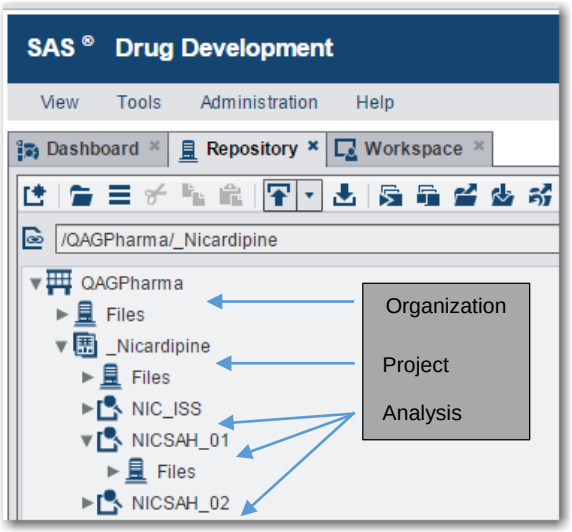
When using SAS Drug Development, it is still possible to use most of these techniques to create an additional copy of programs and data, and some companies continue to use this method. However, there are benefits to avoiding the

copying and storage of programs and data, and this paper offers some additional options that use features in SAS Drug Development to manage snapshots in interesting—and efficient--new ways.

NEW OPTIONS IN SAS DRUG DEVELOPMENT

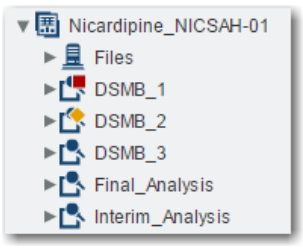
USE PROJECT AND ANALYSIS HIERARCHY

SAS Drug Development provides a hierarchy with three types of contexts: Organization, Project, and Analysis. There is a single Organization, which can be thought of as the root folder for the clinical data repository. Within the Organization, users may create new Projects; and within a Project, users may create new Analyses. Each context has a Files container under which users may create their own folder hierarchy. Within each context, administrative users may manage privileges and permissions independently from the other contexts. The screen shot below illustrates the three SAS Drug Development contexts.



One vision for making use of the Project and Analysis hierarchy is to view the Project as a study, and to view each Analysis as a planned analysis or reporting period for that study. This is not very different from the concept of making and storing a separate copy of a directory structure for a snapshot, except the Analysis hierarchy is planned and created for each snapshot so that work can be done within the Analysis for the planned snapshot, instead of a copy of the data and programs being created after the fact. Each Analysis could be independent, with the full set of programs, data, and macros needed for the Analysis stored within its hierarchy. Or, data and macros applicable to the entire Project could be stored at the Project level and referenced by each Analysis from there.

SAS Drug Development provides the ability to set the State of a context. States include Development, Production and Closed. The default State for a new context is Development, which means that the context's programs are being written, files in the repository are being modified and checked in and out, users are performing actions in the repository, and a user's role and access permissions control the features that the user can access. A user with the appropriate privilege can change the State to Production or Closed. Functionally, setting a context to Production does not change any of the available actions for the context; it is simply an informational indicator that the programs within the context have been through the development cycle and are ready for production use. The Closed state removes the availability of some actions for the context. This state indicates that the context is no longer in use, and the context is locked, and only certain actions are available to users. An Analysis could be set to Closed after delivery of the snapshot, and only opened up again to a Development or Production state if there was a need to modify or reproduce any of its contents. In the screen shot below, the DSMB_1 Analysis has its state set to Closed, the DSMB_2 Analysis has its state set to Production, and the remaining Analyses (and Project) have their states set to Development.



The Closed state prevents the following actions from being performed in the repository:

- Files cannot be checked in or checked out.
- Files cannot be uploaded.
- New containers cannot be created.
- File versioning cannot be changed.
- Object properties cannot be changed.
- Object names cannot be changed.

The Closed state permits the following actions to be performed in the repository:

- Memberships and roles can be changed.
- Access permissions on objects can be changed.
- Files can be placed in the workspace and downloaded.
- Work items and tasks can be worked on.
- Jobs can be run and scheduled. Note that if the output of a job is checked in to a closed Project or Analysis, the job fails.

Using the existing Project and Analysis hierarchy to manage snapshots, with the additional benefit of the State concept, is a simple solution that is a similar concept to the method many companies already use to manage snapshots. However, there are some things to consider when using this method alone. If data and macros that are applicable to all Analyses are stored at the Project level, how will users keep track of the data and macro versions that were used for the snapshot? On the other hand, storing data and macros within every Analysis leads to multiple copies of common files. Also, the required output from one snapshot to the next will be the same or very similar, and the same programs are typically used to create them. When using separate Analyses, all these files will be stored in multiple places and it will be challenging to have traceability between the multiple copies of the same programs and datasets.

CAPTURE AND STORE ALL SNAPSHOT DATA AND PROGRAM METADATA

Instead of maintaining multiple copies of the data and programs in multiple directory structures or Analyses, the rest of this paper explores using a combination of versioned files and SAS Drug Development API SAS macros to both capture snapshot information and retrieve data and files from the snapshot. The SAS Drug Development API SAS Macros provide a variety of functionality for both performing actions within the repository and obtaining information about objects in the repository.

While the rest of this paper describes managing snapshot information within an Analysis, the same technique can apply to a Project, or even a combination of Projects and Analyses. Analysis was chosen to simplify terminology.

The simplest method for managing snapshot information is to version all programs, jobs, and input files, and then store metadata about all of those files at the time of a snapshot delivery. This will work best if all of the related files are versioned; otherwise, if a file that was used during the snapshot is not versioned but is modified, there will be no way to retrieve the precise version used during the snapshot.

The sasdrugdev_getchildren API macro will return, in a SAS dataset, metadata about the objects within the path specified in the macro call. There are three parameters passed to the sasdrugdev_getchildren macro: sdd_path, sas_dsname, and sdd_recursive. The sdd_path is the repository path to the container in SAS Drug Development whose child objects you want to get information about. The sas_dsname is the name of the dataset to store the results, and may include a library specification. The default value for sas_dsname is WORK.SDDGETCHILDREN. The sdd_recursive value indicates the level of recursion the macro should use to check for child objects, and the default value is 0. The following are the valid values for sdd_recursive:

- 0 - Returns metadata for the input container only.
- 1 - Returns metadata for the container and all objects within the container, but not sub containers.
- 2 - Returns metadata for the container, all objects within the container, and one level of sub containers.
- 99 - Returns metadata for the container and all of the sub containers and objects (traverses the entire tree).

The code below shows a call to the sasdrugdev_getchildren macro to obtain metadata about the production listing programs area. It will permanently save the lst_meta dataset to the reports library.

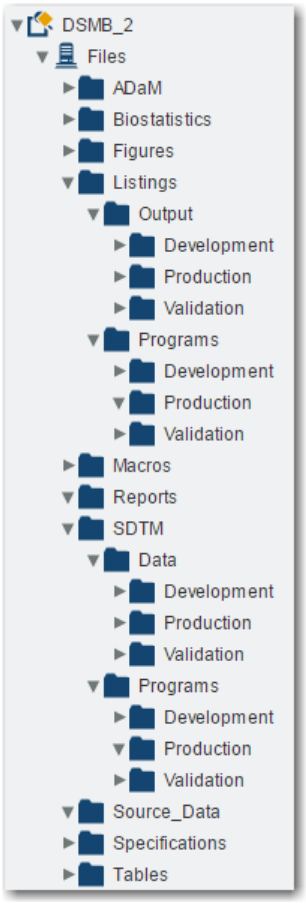
```
libname reports "&_sasws_/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Reports";

%sasdrugdev_getchildren(sdd_path=/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Listings/
Programs/Production,sas_dsname=reports.lst_meta,sdd_recursive=1);
```

The next screenshot shows a partial view of the lst_meta dataset. Notice information about the object type, whether it is a container, the current version, and whether the file is checked out. This can be helpful for managing the snapshot to verify that all of the files are versioned as expected and that no files remain check out at the time of snapshot delivery.

Obs	path	name	objectType	isContainer	description	versionLabel	isVersioned	isCheckedOut	isLocked	isSigned
1	/QAGPharma/N...	Production	sas:folder	1			0	0	0	0
2	/QAGPharma/N...	ae.job	sas:job	0		1.2	1	0	0	0
3	/QAGPharma/N...	ae.sas	sas:sasprogram	0		5.0	1	0	0	0
4	/QAGPharma/N...	ae_listing.sas	sas:sasprogram	0		3.2	1	0	0	0
5	/QAGPharma/N...	AE2_listing.job	sas:job	0		1.2	1	0	0	0

While it may be tempting to use the sdd_recursive=99 option and just run this macro on the Files container under the Analysis, it may take up significant resources and time to run if your Analysis has a large number of objects. It would also produce a dataset that contains metadata about many files that are not the target files of interest for the snapshot. Instead, it is better to use a more deliberate approach to gather the metadata for the snapshot files.

There are many ways to organize a directory hierarchy for a clinical trial, but suppose the following hierarchy was chosen. A similar Output and Programs folder resides under the other output-related folders, such as ADaM, Figures, etc. The files of interest for the snapshot are in the Production folders under each of these.



The sasdrugdev_getchildren macro could be called for each specific folder of interest, and the results from all folders concatenated to create one dataset. The following code could be used to accomplish this.

```
%let analysispath=/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files;

%let folder1=ADaM/Data/Production;
%let folder2=ADaM/Programs/Production;
%let folder3=Figures/Output/Production;
%let folder4=Figures/Programs/Production;
%let folder5=Listings/Output/Production;
%let folder6=Listings/Programs/Production;
%let folder7=SDTM/Data/Production;
%let folder8=SDTM/Programs/Production;
%let folder9=Tables/Output/Production;
%let folder10=Tables/Programs/Production;
%let folder11=Source_Data;
%let folder12=Macros;

libname snapmeta "&_sasws_/QAGPharma/Nicardipine_NICSAH-
01/DSMB_2/Files/Reports";

%macro getsnapshotmetadata;

   %do i=1 %to 12;
      %sasdrugdev_getchildren(sdd_path=&analysispath./&&folder&i,
      sas_dsname=work.children&i, sdd_recursive=1);
   %end;

   data snapmeta.snapshot_dsmb2_20150910;
      set
         %do i=1 %to 12;
            children&i
         %end;
         ;
   run;
%mend getsnapshotmetadata;
%getsnapshotmetadata
```

Storing metadata about the full set of files in the study is straightforward, but it does not necessarily indicate which files were actually used to create output for the snapshot. There could be programs or data that was not used at all. Another interesting consideration is the SAS Drug Development job. A job is required to run a SAS program from the repository. A job requires that you specify the SAS programs to be used, plus the inputs, parameters (if used), and output locations. A job also allows you to specify the version of each input to use. So simply knowing the current version of all files in the Analysis does not tell you whether a job used an earlier version of any of those files as an input. It also doesn't tell you which specific files were used to create the snapshot deliveries.

USE JOB METADATA

One idea to get a more specific list of the files actually used for a snapshot is to capture the metadata from all of the jobs in the Analysis. Conveniently, there are several SAS Drug Development API SAS Macros that return job metadata that can be used for this purpose:

- sasdrugdev_getjobinfo returns metadata about the job itself, such as its name, path, and version, and whether versioning is enabled on output created by the job.
- sasdrugdev_getjobinputs returns metadata about the inputs to the job, including the path, type (file or container), version, and whether inputs of type container are to include subfolders.
- sasdrugdev_getjoboutputs returns metadata about the output paths for the jobs and whether the paths are to include subfolders. Note that outputs for a job are always containers; specific files are not defined.
- sasdrugdev_getjobparameters returns metadata about any parameters defined for the job, including the variable name, its label, its type, and default value.
- sasdrugdev_getjobtasks returns metadata about the SAS programs used in the job, including their location and version.

In general, it would be most useful to know the path and version of the job itself as well as the inputs and SAS programs it uses. A little manipulation is necessary to extract the job name, path, and version from the results of the sasdrugdev_getjobinfo macro.

There are several limitations to this approach. First, the metadata about the inputs may not specify the exact inputs used. Users may simply define an input folder, and they may even specify to include its subfolders. In those cases, you would need to assume that all files in the input folder location(s) were used for the snapshot and capture the

metadata for them using the sasdrugdev_getchildren macro. But this doesn't help narrow down the files used for the snapshot the way one might hope when first considering this option.

Second, if the job uses parameters, the sasdrugdev_getjobparameters returns metadata about the parameters and their defaults, but not the values that were actually used when running the job.

At first glance, using the job metadata may seem like a more precise way to obtain the metadata about the files actually used to create the snapshot delivery data and output. If standard practice at your company is to always specify only the exact inputs and programs, including their versions, for jobs then using the getjob macros could provide a useful set of information. However, if these precise values are not consistently defined in all jobs, the information you can get from the getjob macros will be limited. You also will not be able to trace either the parameter values used or the outputs created by the job.

USE MANIFEST FILE METADATA

When a job is run in SAS Drug Development, a manifest file is created. This manifest file is an XML file (with the .mnf file extension) that stores all of the metadata about what files and parameters were actually used and created when a job was run. This includes the path, version, and file name for the job itself; the specific input files, SAS programs, and output files and their versions; and the parameter values used. To get a completely accurate picture of the exact files used and created for a snapshot, the manifests from all of the jobs that were run can be read to obtain that metadata.

The SAS Drug Development API does not provide tools to directly read the manifest files. However, since they are XML files, they can be read as SAS datasets with the help of an XML map. The metadata provided by a manifest file is rich. Using the sasdrugdev_getchildren macro, you can get a list of all of the manifest files from your snapshot. Then, using clever SAS programming, you can read these manifest files as SAS datasets, retaining the pertinent information to store the snapshot metadata.

The SAS XML Libname Engine can be used to read manifest files, but it will be necessary to use an XML map so that the structure of the manifest can be properly understood by SAS as a set of datasets. The structure of all manifest files in SAS Drug Development are the same; this means you can generate an XML map using one manifest file, and that XML map can then be used when reading all manifest files. The simplest way to do this is to use the automap option in the XML libname. In the code below, you can see the code used to both read an XML file and automatically create an XML map. It specifies the location of the XML file to read as well as the output location to save the XML map. Since this map will be reused frequently for the reading of all manifest files, it makes sense to save this map to a global location.

However, make sure to create the XML map using a manifest file from a job that made use of all of the options within a job, such as creating and using parameters, adding job details, specifying the version of a SAS program, specifying both specific inputs files with versions as well as input locations with "include subfolders" checked, and specifying output locations with "include subfolders" checked. This will make create a manifest file that contains all of the possible information for a manifest, which will allow the automap feature to create a complete XML map. Any manifest files read using the complete XML map will simply have empty datasets if some options were not used when running a job. However, if an incomplete XML map is used to read a manifest file, information from the manifest file would not be captured by the XML libname.

```
filename aemnf "&_sasws_/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Listings/Programs/
Production/ae.mnf";

filename map "&_sasws_/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Specifications/
manifest.map";

libname aemnf xmlv2 automap=replace xmlmap=map;
```

Once you have an XML map generated, you can refer to it whenever reading in other XML files. The code below demonstrates the SAS code to read an XML file using an existing XML map. The code is almost identical, except the automap=replace option has been removed from the libname statement.

```
filename qsmnf "&_sasws_/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/SDTM/Programs/
Production/Create_qs.mnf";

filename map "&_sasws_/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Specifications/
manifest.map";

libname qsmnf xmlv2 xmlmap=map;
```

Using the SAS XML Libname Engine allows SAS to recognize the manifest file as a set of SAS datasets. Five of them are of particular interest and provide all the information about files and parameters actually used when the job was run. The following screenshots from the SAS XML Mapper utility demonstrate partial previews of the contents of the five datasets of interest. First, the job dataset provides the path and version number for the job itself.


XML source | XML Schema source | XMLMap | SAS Code Example | Table view | Contents | Validate | Log

Table: job Row: 1 / 1 Columns: 1 / 10 — SAS formats and informats are not applied to this view.

	manifest_ORDINAL	job_ORDINAL	job_jobPath	job_jobName	job_jobDescription	job_jobVersion
1	1	1	/QAGPharma/Nicardipine/NICSAH_01/Files/Development/SDTM/Programs/Create_qs.job	Create_qs.job		1.0

The file dataset provides the path and version number for the SAS programs used by the job. Notice that the path to the file includes both a prefix and suffix that would need to be parsed out in order to have a path that could be fed to the SAS Drug Development API SAS Macros for the purpose of retrieving the files.

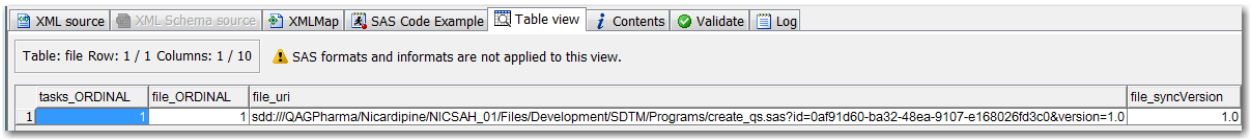

XML source | XML Schema source | XMLMap | SAS Code Example | Table view | Contents | Validate | Log

Table: file Row: 1 / 1 Columns: 1 / 10 — SAS formats and informats are not applied to this view.

	tasks_ORDINAL	file_ORDINAL	file_uri	file_syncVersion
1	1	1	sdd:///QAGPharma/Nicardipine/NICSAH_01/Files/Development/SDTM/Programs/create_qs.sas?id=0af91d60-ba32-48ea-9107-e168026fd3c0&version=1.0	1.0

The file1 and file2 datasets are structured very similarly; file1 describes the inputs used and file2 describes the outputs created. Both provide the path and version of the files, although the version number is part of the file1_uri and file2_uri variables and would need to be parsed out of that variable.

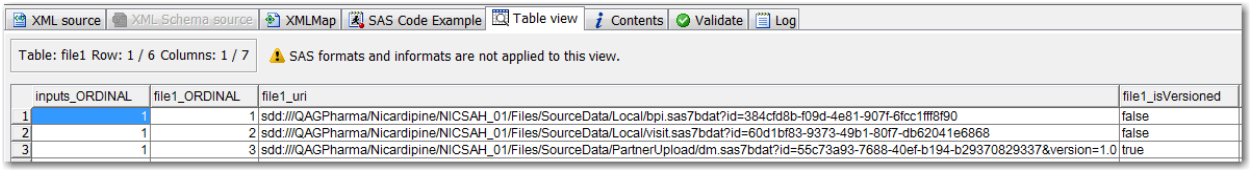

XML source | XML Schema source | XMLMap | SAS Code Example | Table view | Contents | Validate | Log

Table: file1 Row: 1 / 6 Columns: 1 / 7 — SAS formats and informats are not applied to this view.

	inputs_ORDINAL	file1_ORDINAL	file1_uri	file1_isVersioned
1	1	1	sdd:///QAGPharma/Nicardipine/NICSAH_01/Files/SourceData/Local/bpi.sas7bdat?id=384cfd8b-f09d-4e81-907f-6fcc1fff8f90	false
2	1	2	sdd:///QAGPharma/Nicardipine/NICSAH_01/Files/SourceData/Local/visit.sas7bdat?id=60d1bf83-9373-49b1-80f7-db62041e6868	false
3	1	3	sdd:///QAGPharma/Nicardipine/NICSAH_01/Files/SourceData/PartnerUpload/dm.sas7bdat?id=55c73a93-7688-40ef-b194-b29370829337&version=1.0	true

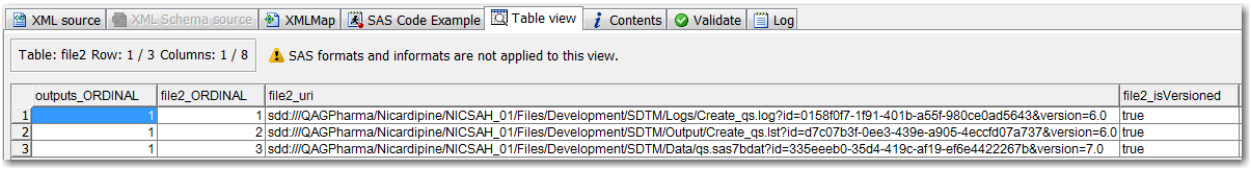

XML source | XML Schema source | XMLMap | SAS Code Example | Table view | Contents | Validate | Log

Table: file2 Row: 1 / 3 Columns: 1 / 8 — SAS formats and informats are not applied to this view.

	outputs_ORDINAL	file2_ORDINAL	file2_uri	file2_isVersioned
1	1	1	sdd:///QAGPharma/Nicardipine/NICSAH_01/Files/Development/SDTM/Logs/Create_qs.log?id=0158f0f7-1f91-401b-a55f-980ce0ad5643&version=6.0	true
2	1	2	sdd:///QAGPharma/Nicardipine/NICSAH_01/Files/Development/SDTM/Output/Create_qs.lst?id=d7c07b3f-0ee3-439e-a905-4eccfd07a737&version=6.0	true
3	1	3	sdd:///QAGPharma/Nicardipine/NICSAH_01/Files/Development/SDTM/Data/qs.sas7bdat?id=335eeeb0-35d4-419c-af19-ef6e4422267b&version=7.0	true

Finally, the parameter dataset provides the values for any parameters used for the job. This information is best kept in a separate dataset from the file metadata; it is useful if you need to rerun a job and determine which parameter values were used when running the original job.

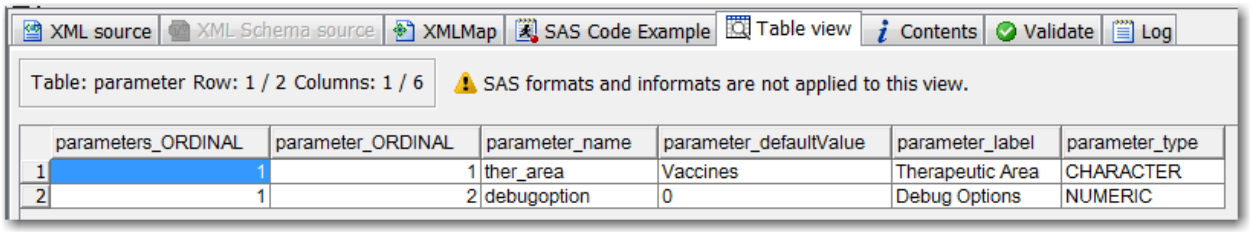

XML source | XML Schema source | XMLMap | SAS Code Example | Table view | Contents | Validate | Log

Table: parameter Row: 1 / 2 Columns: 1 / 6 — SAS formats and informats are not applied to this view.

	parameters_ORDINAL	parameter_ORDINAL	parameter_name	parameter_defaultValue	parameter_label	parameter_type
1	1	1	ther_area	Vaccines	Therapeutic Area	CHARACTER
2	1	2	debugoption	0	Debug Options	NUMERIC

Clearly, some programming is needed to extract and consolidate all the metadata for the job. There are many ways to structure the consolidated metadata. In addition to simply having the path and version number for all the files used, you should specify whether a file is an input, output, or program, as well as relate each of these files to the job (and its version) that used or created the files. The following SAS code could be used to extract and consolidate the metadata from the manifest.

```
filename qsmnf "&_sasws_/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/SDTM/Programs/
Production/Create_qs.mnf";

filename map "&_sasws_/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Specifications/
manifest.map";

libname qsmnf xmlv2 xmlmap=map;

proc sql;
   *Determine input filepath by parsing out sdd://, and removing everything after the ?.;
   *Determine version by finding the version= part of the string, obtaining the text that;
   *comes afterward, and converting that to a numeric value. Leave out inputs that come ;
   *from the work library (begin with work:///);
   create table inoutdata as
      select 'Input' as filetype length=20, scan(tranwrd(file1_uri,"sdd://",""),1,'?')
      as filepath input(tranwrd(scan(file1_uri,2,'&'),"version=",""),best8.) as fileversion
      from qsmnf.file1 where substr(file1_uri,1,4)^='work';
   *Use the same method as above to parse the output information;
```

```
   insert into inoutdata
      select 'Output' as filetype, scan(tranwrd(file2_uri,"sdd://",""),1,'?') as filepath,
    input(tranwrd(scan(file2_uri,2,'&'),"version=",""),best8.) as fileversion
    from qsmnf.file2;
   *Use the same method as above to parse the task (SAS program) information;
   insert into inoutdata
      select 'Task' as filetype, scan(tranwrd(file_uri,"sdd://",""),1,'?') as filepath,
      input(tranwrd(scan(file_uri,2,'&'),"version=",""),best8.) as fileversion
      from qsmnf.file;
   *Obtain the job data;
   create table jobdata as select job_jobName as jobname, job_jobPath as jobpath,
      job_jobVersion as jobversion from qsmnf.job;
   *Use a cross join to insert the job data with every input/output/task file record from;
   *the inoutdata dataset.;
   create table manifestdata as select * from jobdata as lt cross join inoutdata as rt;
quit;
```

The screenshot below shows a view of the dataset created by this code.

Obs	jobname	jobpath	jobversion	filetype	filepath	fileversion
1	Create_qs.job	/QAGPharma/N...	4	Input	/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Source_Data/bpi.sas7bdat	1
2	Create_qs.job	/QAGPharma/N...	4	Input	/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Source_Data/visit.sas7bdat	1
3	Create_qs.job	/QAGPharma/N...	4	Input	/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/SDTM/Data/Production/dm.sas7bdat	1
4	Create_qs.job	/QAGPharma/N...	4	Input	/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Source_Data/ct_qs.sas7bdat	1
5	Create_qs.job	/QAGPharma/N...	4	Input	/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Source_Data/ct_visid.sas7bdat	1
6	Create_qs.job	/QAGPharma/N...	4	Output	/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/SDTM/Programs/Production/Create_qs.log	6
7	Create_qs.job	/QAGPharma/N...	4	Output	/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/SDTM/Programs/Production/Create_qs.lst	6
8	Create_qs.job	/QAGPharma/N...	4	Output	/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/SDTM/Data/Production/qs.sas7bdat	6
9	Create_qs.job	/QAGPharma/N...	4	Task	/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/SDTM/Programs/Production/create_qs.sas	1

ADD ADDITIONAL METADATA

The methods described above use either existing SAS Drug Development API SAS macros or SAS programming to obtain the existing metadata about files used for a snapshot. However, some additional metadata might be useful to facilitate the retrieval process. For example, it could be useful to specify whether a program is a macro, setup file, table program, listing program, or figure program. It could also be useful to specify whether a dataset is source data, SDTM, or ADaM. And it could be useful to specify whether another file is an output file, or even the output file type (listing, figure, table, RTF, PDF, JPG, or whatever designation would be most useful for your organization). This may be useful to help tell the difference between programs and jobs with similar names used for different purposes without having to rely solely on the file path (ae.sas to create the AE SDTM domain, and ae.sas to create the Adverse Event listings). This can also be useful when designing a program to help retrieve the snapshot files. In addition to the ability to retrieve the entire snapshot, the program could allow you to retrieve only one type of program, such as just figure programs. Or you could retrieve only the delivered output files, or only the graph files. The SAS code below shows a simple example of adding metadata to the results of the sasdrugdev_getchildren macro calls before concatenating all the metadata into a single dataset. Using the same folder structure as the example from page 4 of this paper, this code simply uses the first part of each folder's path to add a description to the dataset.

```
%let analysispath=/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files;

%let folder1=ADaM/Data/Production;
%let folder2=ADaM/Programs/Production;
%let folder3=Figures/Output/Production;
%let folder4=Figures/Programs/Production;
%let folder5=Listings/Output/Production;
%let folder6=Listings/Programs/Production;
%let folder7=SDTM/Data/Production;
%let folder8=SDTM/Programs/Production;
%let folder9=Tables/Output/Production;
%let folder10=Tables/Programs/Production;
%let folder11=Source_Data;
%let folder12=Macros;

libname snapmeta "&_sasws_/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Reports";

%macro getsnapshotmetadata;

   %do i=1 %to 12;
      %sasdrugdev_getchildren(sdd_path=&analysispath./&&folder&i, sas_dsname=work.children&i,
sdd_recursive=1);
   %end;
```

```
    data snapmeta.snapshot_dsmb2_20150910;
       set
          %do j=1 %to 12;
             children&j (in=inds&j)
          %end;
          ;

       format filetype $20.;

      if inds1 then filetype=scan("&folder1",1);
      %do k=2 %to 12;
          else if inds&k then filetype=scan("&&folder&k",1);
       %end;
    run;
%mend getsnapshotmetadata;
%getsnapshotmetadata
```

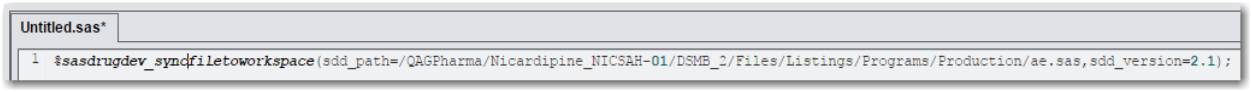
SAVE SNAPSHOT METADATA

There are several different options for obtaining the snapshot metadata. Once this metadata has been retrieved, there should be a single master dataset containing all of the metadata, and it should be stored in a protected location. This metadata is incredibly valuable, and you don't want to lose it! What do you do when you have a new snapshot that you'd like to store metadata for? One option is to just version the snapshot metadata dataset and add a new version to the file (with well-constructed comments describing which snapshot the version pertains to). Another option is to simply store one dataset per snapshot, and name the dataset descriptively so it is clear which snapshot it belongs to.

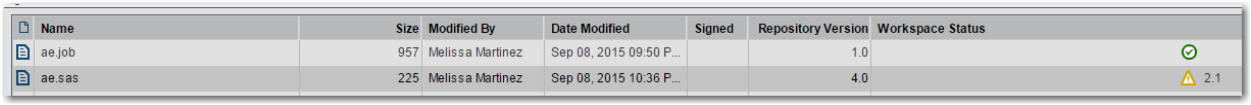
RETRIEVE FILES

This paper has provided a variety of ways of storing snapshot metadata. How do you retrieve part or all of the data and programs for a snapshot? The most straightforward retrieval is to get the version of the file(s) you need into your workspace and use or work on them from there in your workspace. This can be done manually by selecting the file to retrieve, selecting the menu item to get version, and selecting Get Specific Version.

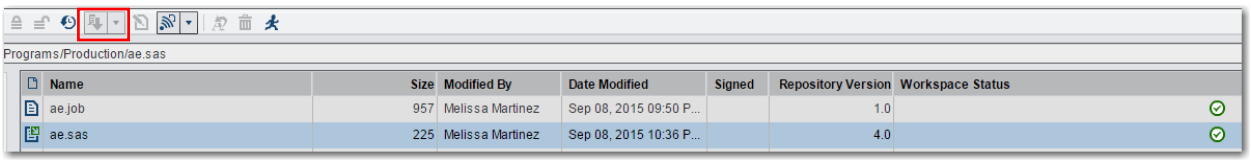
However, if you've stored the snapshot metadata in a SAS dataset, the sasdrugdev_syncfiletoworkspace macro conveniently gets the specific version of the files you need into your workspace. You only need to specify the path to the file (sdd_path) and the version number of the file (sdd_version) in the macro call. The screen shot below demonstrates the syntax for this macro.



Refreshing the repository shows that the version of the file in the workspace is not the same version of the file in the repository, and displays the version of the file in each place.



It gets more complicated if you need to make changes to files and check the updated versions back into the repository. When you perform a Get Version action, whether manually or through the API, SAS Drug Development copies the specific version of the file to your workspace, but it does not check the file out. In order to be able to modify a file in your workspace and place those modifications back into the repository, you must first check out the file so that you can check the updated file back in. But, when you check out a file, it always checks out the latest version of the file. So, if you first get a specific version of a file and then check the file out, the file version in your workspace will be the latest version from the repository. The screenshot below was taken after checking out the ae.sas file after version 2.1 had been obtained in the workspace. The file is now checked out, but version 4.0 is now in the workspace. Notice also that the get version menu item is not active; once a file is checked out, you no longer have the option to get a specific version of that file to your workspace. What can you do?



First, keep in mind that this conundrum is not a problem for outputs you intend to recreate. You may want to get a specific version of an output to your workspace to view it. But if you need to create a new version of it, you can simply

check out the output (which will place the latest version in your workspace), and then run the program or job that will give you updated output in your workspace, and then check the output back into the repository.

WORKAROUND: COPY AND PASTE SAS CODE

Suppose you need to retrieve a previous version of a SAS program, modify it, and save the modified version back to the repository. Get the version you want to your workspace and open the SAS program. Press CTRL-A followed by CTRL-C to copy the code from the SAS editor. If desired, paste the code to a text editor temporarily. Return to the repository and check out the SAS program. Open the checked out SAS program from your workspace, and press CTRL-A followed by CTRL-V to paste the copied code into the SAS editor. Save the program. Now, you can modify the program in your workspace, save it, and check it back into the repository to create a new version of it.

WORKAROUND: RENAME FILES IN WORKSPACE

The previously described technique works for SAS programs, but not for other file types like documents, Excel files, or SAS Drug Development jobs. One solution for these files is to first get the version of the file(s) you want to your workspace. Then, simply copy and paste the files in place in the workspace; SAS Drug Development will automatically prepend each file name with “Copy x of”, where x is a number.

Name	Size	Date Modified	Repository Version	Workspace Status
ae.job	957	Sep 08, 2015 09:50 P...	1.0	✓
ae.sas	225	Sep 08, 2015 10:35 P...	4.1	⚠ 2.1
AE2_listing.job	1110	Sep 08, 2015 09:50 P...	1.0	✓
ae_listing.sas	228	Sep 08, 2015 10:35 P...	3.0	⚠ 2.0
Copy 1 of ae.sas	225	Sep 09, 2015 01:12 A...		
Copy 1 of ae_listing.sas	228	Sep 09, 2015 01:12 A...		
Copy 1 of subjlisting.sas	237	Sep 09, 2015 01:12 A...		
print_ae.sas	229	Sep 08, 2015 10:35 P...	2.0	✓
runsubjinfolisting.job	1081	Sep 08, 2015 09:50 P...	1.0	✓
subjlisting.sas	237	Sep 08, 2015 10:35 P...	2.0	⚠ 1.2
test_code.job	638	Sep 08, 2015 09:50 P...	1.0	✓
test_code.sas	422	Sep 08, 2015 09:50 P...	1.0	✓

Next, check the files out from the repository, which will result in two copies of the desired file(s) in your workspace, the one just checked out, and the desired version that has been renamed with “Copy x of” prepended to the file name.

Name	Size	Date Modified	Repository Version	Workspace Status
ae.job	957	Sep 08, 2015 09:50 P...	1.0	✓
ae.sas	225	Sep 09, 2015 01:11 A...	4.1	✓
AE2_listing.job	1110	Sep 08, 2015 09:50 P...	1.0	✓
ae_listing.sas	228	Sep 08, 2015 10:36 P...	3.0	✓
Copy 1 of ae.sas	225	Sep 09, 2015 01:12 A...		
Copy 1 of ae_listing.sas	228	Sep 09, 2015 01:12 A...		
Copy 1 of subjlisting.sas	237	Sep 09, 2015 01:12 A...		
print_ae.sas	229	Sep 08, 2015 10:35 P...	2.0	✓
runsubjinfolisting.job	1081	Sep 08, 2015 09:50 P...	1.0	✓
subjlisting.sas	237	Sep 08, 2015 10:36 P...	2.0	✓
test_code.job	638	Sep 08, 2015 09:50 P...	1.0	✓
test_code.sas	422	Sep 08, 2015 09:50 P...	1.0	✓

Then, delete the checked-out version of the files you copied.

Name	Size	Date Modified	Repository Version	Workspace Status
ae.job	957	Sep 08, 2015 09:50 P...	1.0	✓
AE2_listing.job	1110	Sep 08, 2015 09:50 P...	1.0	✓
Copy 1 of ae.sas	225	Sep 09, 2015 01:12 A...		
Copy 1 of ae_listing.sas	228	Sep 09, 2015 01:12 A...		
Copy 1 of subjlisting.sas	237	Sep 09, 2015 01:12 A...		
print_ae.sas	229	Sep 08, 2015 10:35 P...	2.0	✓
runsubjinfolisting.job	1081	Sep 08, 2015 09:50 P...	1.0	✓
test_code.job	638	Sep 08, 2015 09:50 P...	1.0	✓
test_code.sas	422	Sep 08, 2015 09:50 P...	1.0	✓

If you return to the repository, you will find that the Workspace Status indicator is a red x, indicating the file is missing from the workspace.

Name	Size	Modified By	Date Modified	Signed	Repository Version	Workspace Status
ae.job	957	Melissa Martinez	Sep 08, 2015 09:50 P...		1.0	
ae.sas	225	Melissa Martinez	Sep 09, 2015 01:11 A...		4.1	4.1
AE2_listing.job	1110	Melissa Martinez	Sep 08, 2015 09:50 P...		1.0	
ae_listing.sas	228	Melissa Martinez	Sep 08, 2015 10:36 P...		3.0	3.0
print_ae.sas	229	Melissa Martinez	Sep 08, 2015 10:35 P...		2.0	
runsubjinfolisting.job	1081	Melissa Martinez	Sep 08, 2015 09:50 P...		1.0	
subjlisting.sas	237	Melissa Martinez	Sep 08, 2015 10:36 P...		2.0	2.0
test_code.job	638	Melissa Martinez	Sep 08, 2015 09:50 P...		1.0	
test_code.sas	422	Melissa Martinez	Sep 08, 2015 09:50 P...		1.0	

Finally, rename the remaining files to remove the “Copy x of” extension. Returning to the repository, you will find that the Workspace Status indicator is a yellow warning indicating the workspace version is different from that of the repository version.

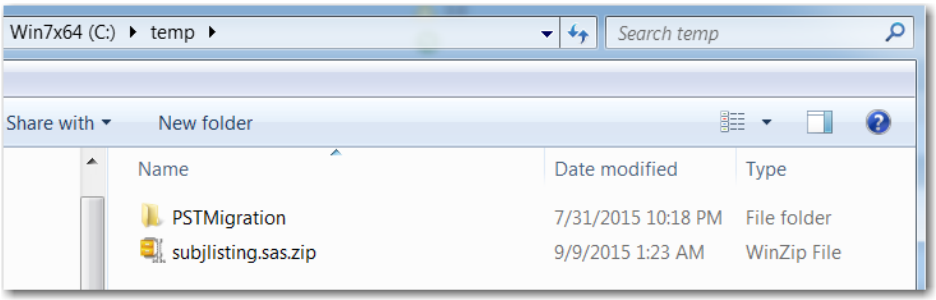
Name	Size	Modified By	Date Modified	Signed	Repository Version	Workspace Status
ae.job	957	Melissa Martinez	Sep 08, 2015 09:50 P...		1.0	
ae.sas	225	Melissa Martinez	Sep 09, 2015 01:11 A...		4.1	4.1
AE2_listing.job	1110	Melissa Martinez	Sep 08, 2015 09:50 P...		1.0	
ae_listing.sas	228	Melissa Martinez	Sep 08, 2015 10:36 P...		3.0	3.0
print_ae.sas	229	Melissa Martinez	Sep 08, 2015 10:35 P...		2.0	
runsubjinfolisting.job	1081	Melissa Martinez	Sep 08, 2015 09:50 P...		1.0	
subjlisting.sas	237	Melissa Martinez	Sep 08, 2015 10:36 P...		2.0	2.0
test_code.job	638	Melissa Martinez	Sep 08, 2015 09:50 P...		1.0	
test_code.sas	422	Melissa Martinez	Sep 08, 2015 09:50 P...		1.0	

WORKAROUND: DOWNLOAD AND UPLOAD

The final solution discussed here is downloading and uploading files from SAS Drug Development. This technique should not be used for data, if your company has a policy against downloading data out of the SAS Drug Development environment. Just like the renaming solution, get the version of the files you want to your workspace.

Name	Size	Date Modified	Repository Version	Workspace Status
ae.job	957	Sep 09, 2015 01:20 A...	1.1	
ae.sas	225	Sep 08, 2015 10:36 P...	4.2	3.0
AE2_listing.job	1110	Sep 09, 2015 01:20 A...	1.1	
ae_listing.sas	228	Sep 08, 2015 10:35 P...	3.1	2.0
print_ae.sas	229	Sep 09, 2015 01:20 A...	2.1	
runsubjinfolisting.job	1081	Sep 09, 2015 01:20 A...	1.1	
subjlisting.sas	237	Sep 08, 2015 10:36 P...	2.1	1.3
test_code.job	638	Sep 09, 2015 01:20 A...	1.1	
test_code.sas	422	Sep 09, 2015 01:20 A...	1.1	

Then download the out-of-sync files from your workspace to a location on your local PC. In the case of a PC-based files like an Excel file or document, they would need to be updated from your local PC anyway; SAS does not provide a built-in editor for non-SAS files. Note that if you select more than one file to download, it will download as a zip file.



If needed for non-SAS files, update the file on your local PC and save it. Next, check the files out of the repository.

Name	Size	Date Modified	Repository Version	Workspace Status
ae.job	957	Sep 09, 2015 01:20 A..	1.1	✓
ae.sas	225	Sep 09, 2015 01:20 A..	4.2	✓
AE2_listing.job	1110	Sep 09, 2015 01:20 A..	1.1	✓
ae_listing.sas	228	Sep 09, 2015 01:20 A..	3.1	✓
print_ae.sas	229	Sep 09, 2015 01:20 A..	2.1	✓
runsubjinfolisting.job	1081	Sep 09, 2015 01:20 A..	1.1	✓
subjlisting.sas	237	Sep 09, 2015 01:20 A..	2.1	✓
test_code.job	638	Sep 09, 2015 01:20 A..	1.1	✓
test_code.sas	422	Sep 09, 2015 01:20 A..	1.1	✓

Next, upload the copy of the files from your local PC back to your workspace.

Name	Size	Date Modified	Repository Version	Workspace Status
ae.job	957	Sep 09, 2015 01:20 A..	1.1	✓
ae.sas	225	Sep 09, 2015 01:26 A..	4.2	⚠ 4.2
AE2_listing.job	1110	Sep 09, 2015 01:20 A..	1.1	✓
ae_listing.sas	228	Sep 09, 2015 01:26 A..	3.1	⚠ 3.1
print_ae.sas	229	Sep 09, 2015 01:20 A..	2.1	✓
runsubjinfolisting.job	1081	Sep 09, 2015 01:20 A..	1.1	✓
subjlisting.sas	237	Sep 09, 2015 01:26 A..	2.1	⚠ 2.1
test_code.job	638	Sep 09, 2015 01:20 A..	1.1	✓
test_code.sas	422	Sep 09, 2015 01:20 A..	1.1	✓

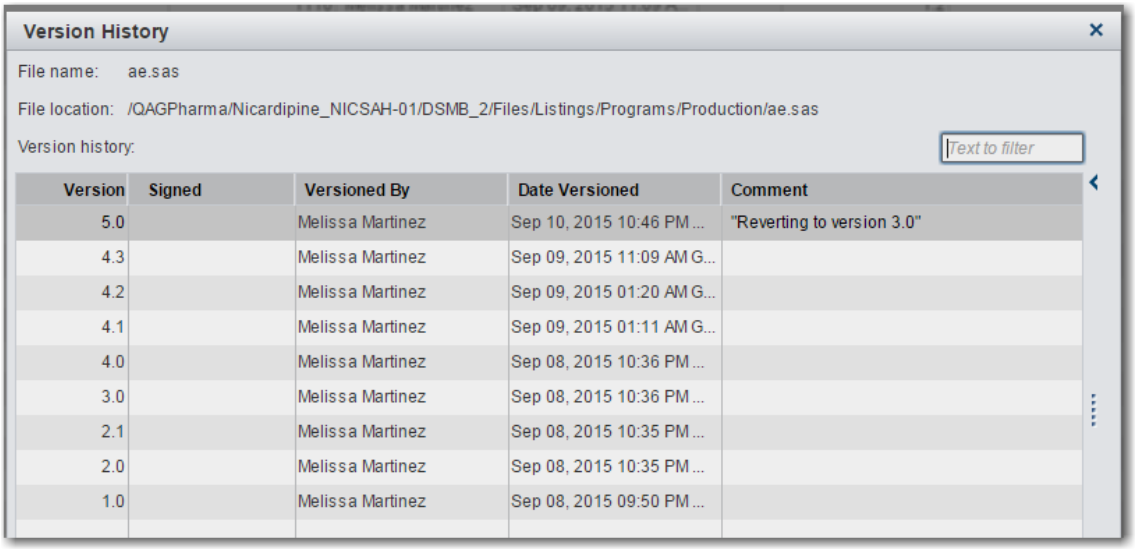
If the files are SAS jobs or programs, you may then update the files in your workspace. Next, check the files back into the repository to create a new version of the files.

The solutions discussed so far for retrieving previous versions of files so that they can be modified and saved back to the repository are very manual processes. There is one more technique that uses the SAS Drug Development API Macros to automate the download and upload of files directly from and to the repository to save the desired previous version of the file as the newest version of the file. From there, the file can be checked out to user workspaces, worked on, and checked back in to the repository like a regular file. This technique requires PC SAS and the SAS Drug Development API Macros version 1.4, as the macros used in this technique only work from outside of SAS Drug Development. For more information on downloading and installing the SAS Drug Development API Macros, please see Installation Note 46609: SAS® Drug Development 4 - Accessing Java API all versions client and macro downloads.

The sasdrugdev_downloadfile macro allows you to download a file from the SAS Drug Development repository to a specified path on your local PC. The sasdrugdev_updatefile macro allows you to upload a new copy of a file to the SAS Drug Development repository. The following SAS code will download the current version of a file from the SAS Drug Development repository and then upload it back to the repository as a new major version.

```
%sasdrugdev_login(sdd_url=https://sdddemo.sas.com, sdd_userid=<userid>, sdd_password=<pass>);

%sasdrugdev_downloadfile(sdd_path=/QAGPharma/Nicardipine_NICSAH-01/DSMB_2/Files/Listings/
Programs/Production/ae.sas, sdd_version=3.0, local_path=C:\temp\ae.sas);

%sasdrugdev_updatefile(local_path=C:\temp\ae.sas, sdd_path=/QAGPharma/Nicardipine_NICSAH-
01/DSMB_2/Files/Listings/Programs/Production/ae.sas, sdd_comment="Reverting to version 3.0",
sdd_version=MAJOR);

%sasdrugdev_logout;
```

Returning to the SAS Drug Development repository shows a new version of the file has been created. The screenshot below shows the version history for this file.



Version	Signed	Versioned By	Date Versioned	Comment
5.0		Melissa Martinez	Sep 10, 2015 10:46 PM ...	"Reverting to version 3.0"
4.3		Melissa Martinez	Sep 09, 2015 11:09 AM G...	
4.2		Melissa Martinez	Sep 09, 2015 01:20 AM G...	
4.1		Melissa Martinez	Sep 09, 2015 01:11 AM G...	
4.0		Melissa Martinez	Sep 08, 2015 10:36 PM ...	
3.0		Melissa Martinez	Sep 08, 2015 10:36 PM ...	
2.1		Melissa Martinez	Sep 08, 2015 10:35 PM ...	
2.0		Melissa Martinez	Sep 08, 2015 10:35 PM ...	
1.0		Melissa Martinez	Sep 08, 2015 09:50 PM ...	

CONCLUSION

SAS Drug Development is a clinical data repository that offers a rich set of features for managing the data, programs, and other files related files to clinical trials. Results and output may be delivered as snapshots at several different points throughout the lifecycle of a clinical trial, and it is important to be able to retrieve and reproduce the data, programs, and results that generated each of those snapshots. The most common technique for saving snapshot information is to make an additional copy of the data, programs, and results, and store the copy in an additional location. This paper describes the unique features in SAS Drug Development that enable more efficient snapshot management, including using the Project and Analysis hierarchy to separate snapshots, and methods for gathering and storing metadata about the files used for the snapshot. This paper also described several methods for retrieving files in order to recreate or modify previously delivered files. With a combination of versioning, metadata retrieval, and API SAS Macro capabilities, SAS Drug Development provides some interesting and novel ways of managing snapshot metadata more easily.

REFERENCES

SAS Drug Development 4.5 User Guide
SAS Drug Development Macro API Release 1.4 User Guide

ACKNOWLEDGMENTS

Thank you to the SAS Solutions OnDemand Health and Life Sciences Consulting Team for being such a hard-working, dedicated, and supportive team. I'm continuously thankful to work among you. Thank you to Jensen Mabe for dependable, intelligent, and cheerful copyediting and technical writing support. As always, I remain grateful for Angela Lightfoot, an extraordinary manager who provides me with the right balance of challenge and support.

CONTACT INFORMATION

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

Melissa R. Martinez
SAS Institute
100 SAS Campus Drive
Cary, North Carolina 27513
Email: Melissa.Martinez@sas.com
Web: https://communities.sas.com/community/support-communities/sas-drug-development