Paper 1067-2017

An Introduction to the Improved SAS® Data Integration Studio Batch Deployment Utility on UNIX

Jeff Dyson, The Financial Risk Group

ABSTRACT

Interactively redeploying SAS® Data Integration Studio jobs can be a slow and tedious process. The updated batch deployment utility gives the ETL Tech Lead a more efficient and repeatable method for administering batch jobs. This improved tool became available in SAS Data Integration Studio 4.901.

INTRODUCTION

There are many benefits of using SAS Data Integration Studio. One of its fundamental functions is to generate code. Metadata – data about data – and transformations work together to produce SAS code that populates data warehouses and data marts. With SAS' powerful analytics, this data is essential for solving business problems.

SAS Data Integration Studio is rarely used for ad-hoc data investigations or what if analysis. Instead, it's more suitable for repeatable extract, transform, and load processing that runs in a batch environment on daily, weekly, or monthly intervals.

The purpose of this paper is to introduce the revised batch deployment utility which moves the generated code from SAS Data Integration Studio to the operating system so it is available for batch processing.

THE JOB, GENERATED CODE, AND THE DEPLOY PROCESS

After data modeling and mapping exercises are complete, developers are typically given project documentation that serves as a roadmap for moving data from source systems to a data warehouse. When broken down into smaller units of work, individual jobs can be as simple as a SQL based Extract or Join followed by a Table Loader that populates a SAS dataset or database table. The developer can use the SAS Data Integration Studio drag and drop interface to build the job. Figure 1 is a SAS Data Integration Studio job.

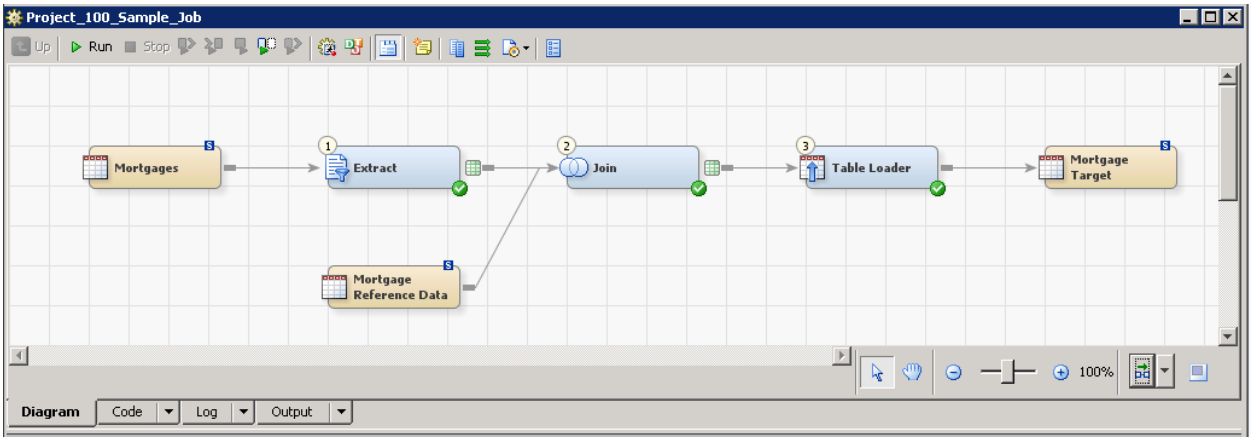

Figure 1. SAS® Data Integration Studio Job

GENERATED CODE

Once a job is built, the user can view the generated code by selecting the Code tab in the lower left hand corner of the job canvas. Figure 2 shows the generated code as well as the Code tab.

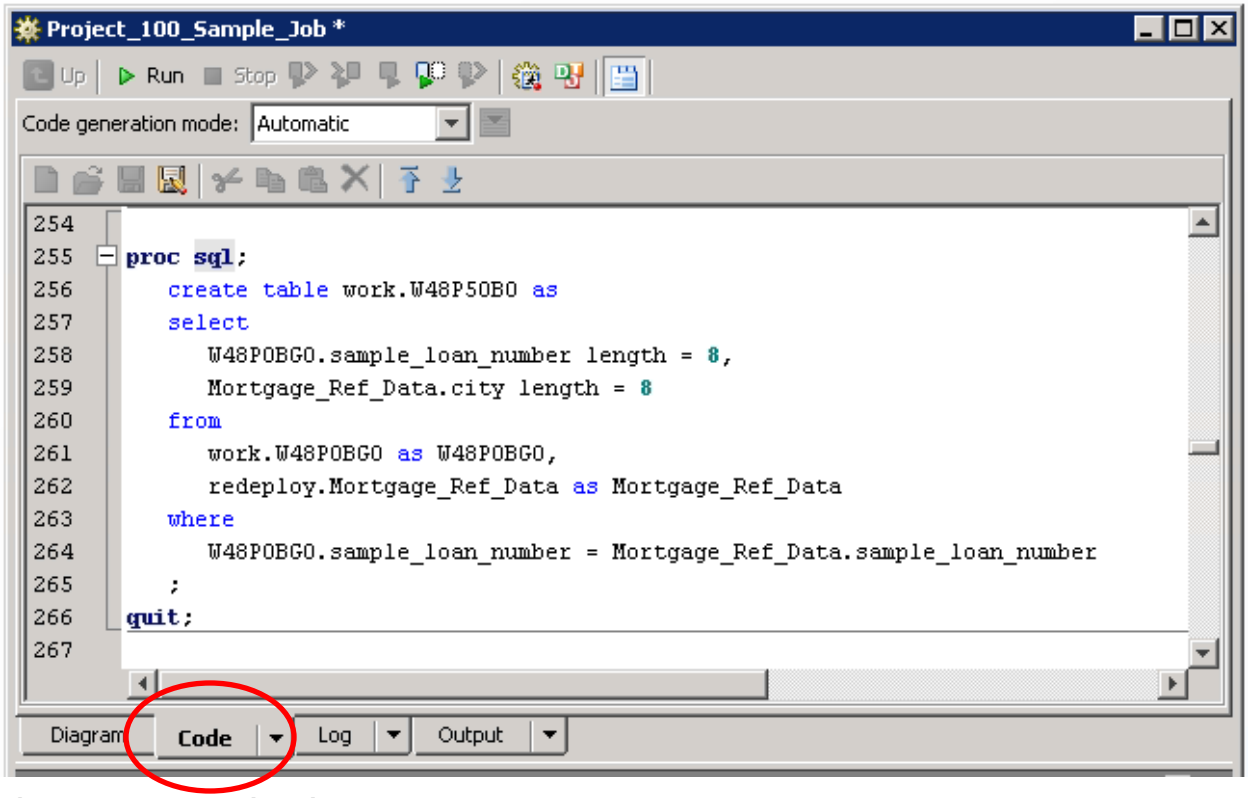

Figure 2. Generated Code

DEPLOYING

In order to run a job in batch it must be deployed. Deploying creates a deployed job metadata object which makes the job available to flows in the Schedule Manager plug-in in SAS® Management Console. Deploying also writes the generated code to a file on the application server so it can be executed using Platform Load Sharing Facility (LSF), an operating system scheduler, or UNIX script.

To manually deploy a job, right click on the job and use the Scheduling menu as seen in Figure 3. Figure 4 illustrates the parameters that are required.

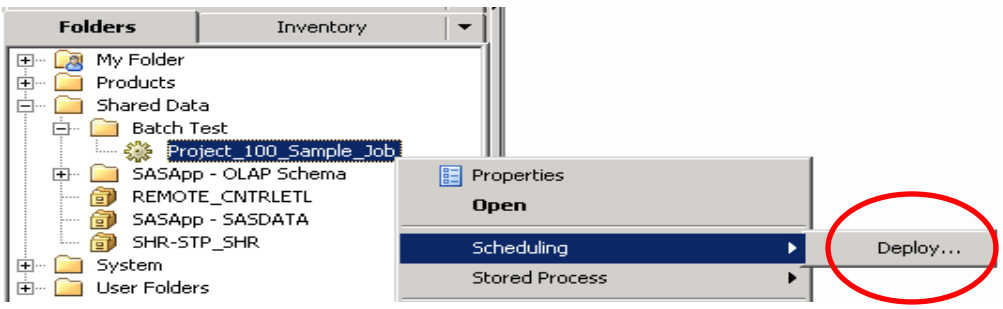

Figure 3. Manually Deploying a Job

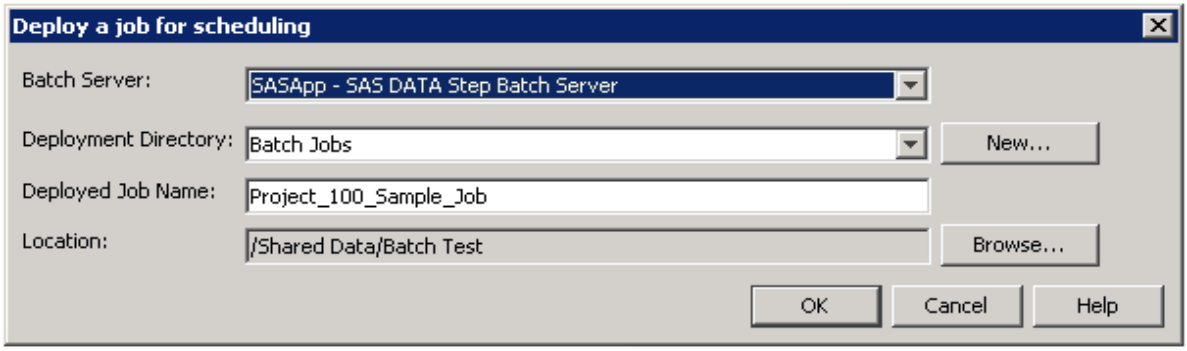

Figure 4. Deployed Job Object Parameters

REDEPLOYING

As a project proceeds, it is not uncommon for a developer to receive new business requirements which lead to logic changes in jobs that were previously deployed. Implementing these updates will create a new version of the generated code which must be redeployed to the application server so it is available to batch schedulers or scripts. A job can be redeployed as often as needed. The redeploy process will overwrite the old deployed code but does not create a new deployed job metadata object.

To manually redeploy a job, right click on the job and select Redeploy with the Scheduling menu as seen in Figure 5. Figure 6 shows that the parameters from the initial deploy are used to redeploy the code.

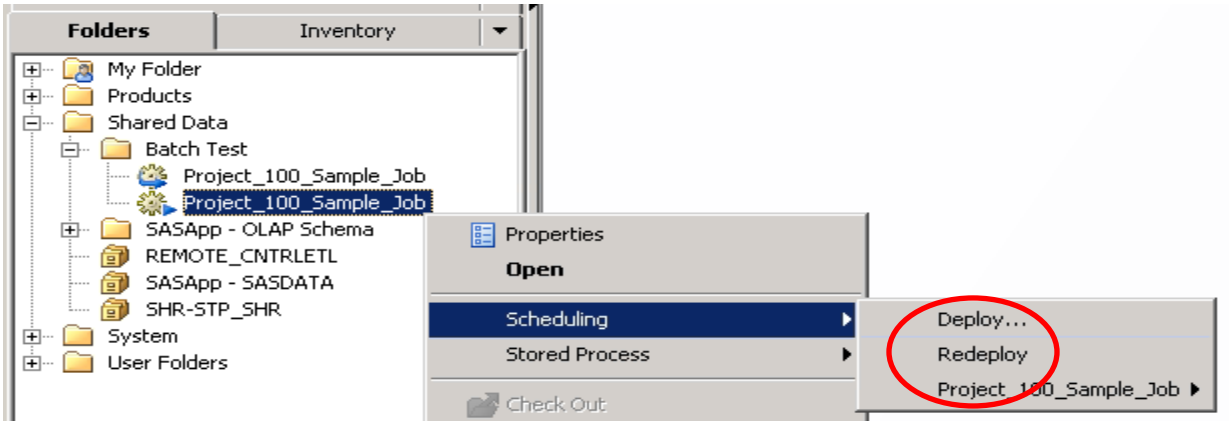

Figure 5. Manually Redeploying a Job

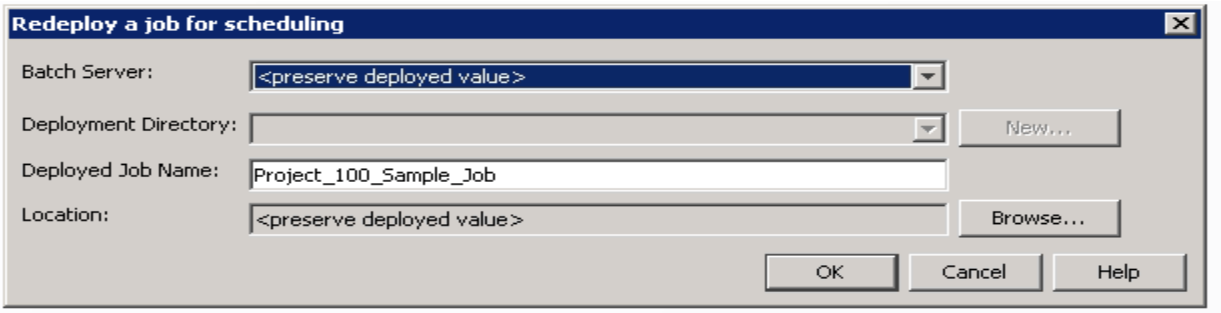

Figure 6. Redeploy Job Object Parameters

AUTOMATING THE DEPLOY PROCESS

Manually deploying and redeploying a small number of jobs is not a difficult or time consuming task. But imagine a project with fifty to one hundred or more jobs. In this case, consider using the revised command line batch deployment utility that became available with SAS Data Integration Studio 4.901.

THE BASICS

In the UNIX environment, the batch deployment executable is named "DeployJobs" and is installed under the /path/to/install/SASDataIntegrationStudioServerJARs/4.8 directory. There are two groups of options that must be used to run DeployJobs. The first set is considered Connection Options and is used to determine how to connect to the SAS metadata server that stores SAS Data Integration Studio jobs. Table 1 lists these options.

Option	Description
host	Metadata server hostname
port	Metadata server port
user	Username
password	Password

Table 1. Metadata Server Connection Options

Note that if the user has a metadata connection profile defined on the UNIX host that is being used to run DeployJobs, the PROFILE option can replace these four parameters.

The second group of options controls the deploy or redeploy process. The following options can be used to create a script that will deploy a single job.

Option	Description
deploytype	DEPLOY
objects	Object(s) to deploy
sourcedir	Unix directory that will stored the generated code
deploymentdir	Unix directory that will stored the generated code
metarepository	Name of metadata repository
appservername	Application server name
servermachine	Hostname running application server
serverport	Application server port
serverusername	User id used to connect to application server
serverpassword	Password used to connect to application server
batchserver	Batch server
log	UNIX filename containing DeployJobs output

Table 2. Basic Deployment Options

These options are documented in the SAS Data Integration Studio User's Guide in the section titled 'Using a Command Line to Deploy Jobs'.

HELPFUL TIPS AND A SCRIPT TO DEPLOY ONE JOB

When using DeployJobs for the first time, a good approach is to create a script that will deploy a single job. The following script uses the OBJECTS option to deploy a job named Project_100_Sample_Job.

Recall that code is written to the application server when a job is deployed. During a manual deploy, the code is written to the deployment directory – a single location - as defined in SAS Management Console or SAS Data Integration Studio. However the DeployJobs utility requires two parameters that point to UNIX directories. These options are SOURCEDIR and DEPLOYMENTDIR and from a user's perspective one would seem to be unnecessary. In reality, two options are required because DeployJobs uses a common API made available by the SAS Management Console Schedule Manager. Regardless, the user can point both options to a single UNIX directory that will store the generated code.

Notice in this example the unrestricted user id connects to the metadata server while another user id starts the workspace server. The same id can be used for both options as long as it is not an internal SAS account (like sasadm@saspw) and it has the necessary metadata and target directory permissions. The id specified in the SERVERUSERNAME option will own the file which contains the generated code on the operating system. It is also important to note these credentials may be used for database connections in the generated code. To avoid this issue, the USEAUTHDOMAIN option can control whether code is deployed with USER= and PASSWORD= or with AUTHDOMAIN=. To take advantage of an authentication domain, it must be defined in metadata with credentials that have access to the respective database management system.

The LOG parameter can be used to capture standard output produced by the command. Even when using this option, standard output is written to the terminal. If no standard output is desired, the user can take advantage of UNIX redirection (i.e. > and >>) to write output to a file.

Lastly, while the user could enter the entire DeployJobs command at a UNIX prompt, it is much easier to create a script. By using a pound sign (#) for comments and a backward slash (\) at the end of each line, a script allows the user to create a documented and repeatable process. The following script illustrates how to deploy a single job named Project_100_Sample_Job:

```
 # Deploy one job
/path/to/sas94/install/SASDataIntegrationStudioServerJARs/4.8/DeployJobs \
-host <metadata_server_host> -port 8561 \
-user sasadm@saspw -password "XXXXXXXXX" \
-deploytype DEPLOY \
-objects "/Shared Data/Batch Test/Project_100_Sample_Job" \
-sourcedir /path/to/scripts/batch_redeploy/deployed_jobs \
-deploymentdir /path/to/scripts/batch_redeploy/deployed_jobs \
-metarepository Foundation \
-appservername SASApp \
-servermachine <application_server_host> \
-serverport 8591 \
-serverusername batch_id \
-serverpassword "XXXXXXXXX" \
-batchserver "SASApp - SAS DATA Step Batch Server" \
-log /path/to/scripts/batch_redeploy/deploy.log
```

In this example, the script is named deploy.sh and is executed by typing "./deploy.sh" at the UNIX prompt. When run, the following output is produced:

```
[/path/to/scripts/batch_redeploy]$./deploy.sh
INFO  Initializing Job Deployment environment.
INFO  A deployment type of DEPLOY has been specified.
```

```
INFO  Job 'Project_100_Sample_Job' in '/Shared Data/Batch Test' is being processed.
11:43:26 WARN  DISVersion for job: Project_100_Sample_Job:A5GE6AHJ.C80001BD was set to 0.0. Workspace's
version number is: 0.0
INFO  Job 'Project_100_Sample_Job' was successfully deployed.
INFO  Batch execution complete.
[/path/to/scripts/batch_redeploy]$
```

The script creates generated code for the job in the SOURCEDIR and DEPLOYMENTDIR UNIX directory:

```
[/path/to/scripts/batch_redeploy/deployed_jobs]$ls -ltr
total 16
-rw-rw-r-- 1 batch_id 14079 Feb 26 11:43 Project_100_Sample_Job.sas
[/path/to/scripts/batch_redeploy/deployed_jobs]$
```

By default, the deployed job metadata object is written to the same metadata folder that stores the job. Figure 7 shows the deployed job metadata object.

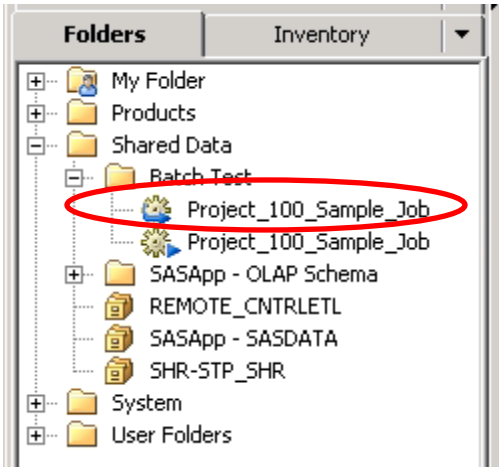

Figure 7. Deployed Job Metadata

DEPLOYING MULTIPLE OBJECTS

Using a script to deploy a single object isn't necessary since the job can be deployed manually with little effort. But as previously mentioned, the DeployJobs utility adds more value when multiple objects must be deployed. One way to deploy multiple objects is to simply list each job in the OBJECTS option. The following script illustrates how to deploy multiple jobs:

```
 # Deploy multiple jobs
/path/to/sas94/install/SASDataIntegrationStudioServerJARs/4.8/DeployJobs \
-host <metadata_server_host> -port 8561 \
-user sasadm@saspw -password "XXXXXXXXX" \
-deploytype DEPLOY \
-objects "/Shared Data/Batch Test/Project_100_Sample_Job" \
         "/Shared Data/Batch Test/Project_110_Sample_Job" \
         "/Shared Data/Batch Test/Project_120_Sample_Job" \
-sourcedir /path/to/scripts/batch_redeploy/deployed_jobs \
-deploymentdir /path/to/scripts/batch_redeploy/deployed_jobs \
-metarepository Foundation \
-appservername SASApp \
-servermachine <application_server_host> \
```

```
-serverport 8591 \
-serverusername batch_id \
-serverpassword "XXXXXXXXX" \
-batchserver "SASApp - SAS DATA Step Batch Server" \
-log /path/to/scripts/batch_redeploy/deploy.log
```

In this example, the script creates three generated code files in the UNIX directory defined by the SOURCEDIR and DEPLOYMENTDIR options.

DEPLOYING AN ENTIRE FOLDER AND THE FOLDER OPTION

If all the jobs in a metadata folder need to be deployed, change the OBJECTS option so that it points to that folder. To be clear, in this scenario the OBJECTS option would not reference a job by name.

This version of the script also introduces the FOLDER option. The FOLDER option controls where deployed job metadata objects are created. One simple convention is to store these objects in a metadata folder named Deployed Jobs. The following script illustrates how to deploy all of the jobs in a folder while also creating deployed job metadata in a named folder:

```
# Deploy all jobs
/path/to/sas94/install/SASDataIntegrationStudioServerJARs/4.8/DeployJobs \
-host <metadata_server_host> -port 8561 \
-user sasadm@saspw -password "XXXXXXXXX" \
-deploytype DEPLOY \
-objects "/Shared Data/Batch Test" \
-folder "/Shared Data/Batch Test/Deployed Jobs" \
-sourcedir /path/to/scripts/batch_redeploy/deployed_jobs \
-deploymentdir /path/to/scripts/batch_redeploy/deployed_jobs \
-metarepository Foundation \
-appservername SASApp \
-servermachine <application_server_host> \
-serverport 8591 \
-serverusername batch_id \
-serverpassword "XXXXXXXXX" \
-batchserver "SASApp - SAS DATA Step Batch Server" \
-log /path/to/scripts/batch_redeploy/deploy.log
```

In this example, DeployJobs generates code for all of the jobs in the Batch Test metadata folder. As expected, the files are created in the UNIX directory specified in the SOURCEDIR and DEPLOYMENTDIR options:

```
[/path/to/scripts/batch_redeploy/deployed_jobs]$ls
Project_100_Sample_Job.sas  Project_130_Sample_Job.sas  Project_160_Sample_Job.sas
Project_190_Sample_Job.sas  Project_110_Sample_Job.sas  Project_140_Sample_Job.sas
Project_170_Sample_Job.sas  Project_200_Sample_Job.sas  Project_120_Sample_Job.sas
Project_150_Sample_Job.sas  Project_180_Sample_Job.sas
[/path/to/scripts/batch_redeploy/deployed_jobs]$
```

Figure 8 shows that metadata folders can be better organized when using the FOLDER option.

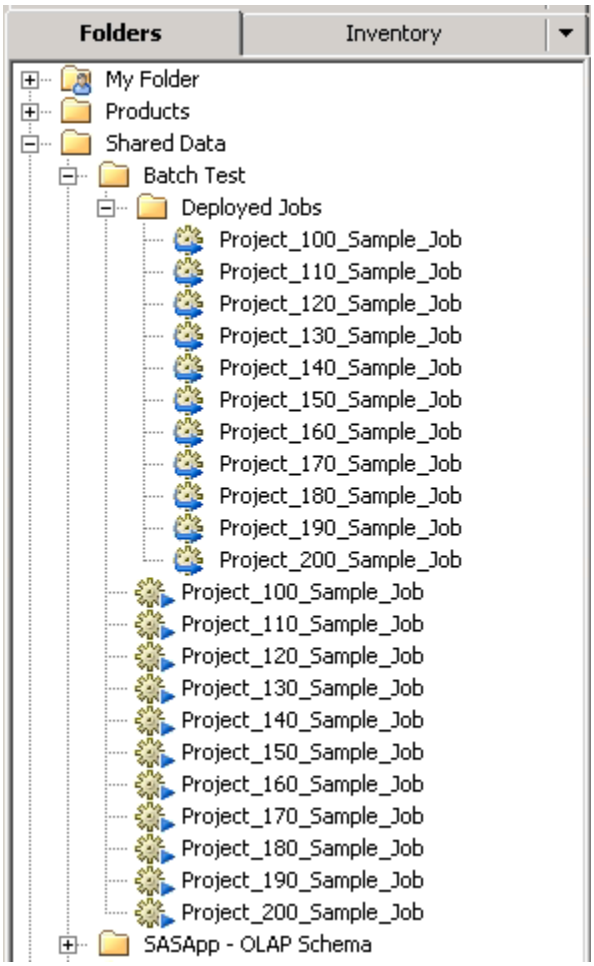

Figure 8. Organize Metadata with the folder option

THE SINCE OPTION

The SINCE option can be used to determine which objects to deploy based on the date and time of the last modified date. Consider a scenario where one job was last modified on 16Feb2017 and the remaining jobs in the metadata folder were updated on 27Feb2017. By specifying a date of 20Feb2017, the DeployJobs utility will only deploy objects that were modified after 20Feb2017. The user can also specify a time, but 12:00:00 am is the default if no time is provided. The following script shows how to use the SINCE option:

```
# Deploy with the since option
/path/to/sas94/install/SASDataIntegrationStudioServerJARs/4.8/DeployJobs \
-host <metadata_server_host> -port 8561 \
-user sasadm@saspw -password "XXXXXXXXX" \
-deploytype DEPLOY \
-objects "/Shared Data/Batch Test" \
-folder "/Shared Data/Batch Test/Deployed Jobs" \
-sourcedir /path/to/scripts/batch_redeploy/deployed_jobs \
-deploymentdir /path/to/scripts/batch_redeploy/deployed_jobs \
-metarepository Foundation \
-appservername SASApp \
```

```
-servermachine <application_server_host> \
-serverport 8591 \
-serverusername batch_id \
-serverpassword "XXXXXXXXX" \
-batchserver "SASApp - SAS DATA Step Batch Server" \
-log /path/to/scripts/batch_redeploy/deploy.log \
-since 02/20/2017
```

Because one job was last modified on 16Feb2017, a date prior to the SINCE option date of 20Feb2017, it would not be deployed.

When adding options to the end of the script, remember to add a backward slash to the end of the line which was previously the last line in the script.

AUTOMATING THE REDEPLOY PROCESS

In most cases a SAS Data Integration Studio job is deployed one time so that it is available for batch processing. However if business requirements lead to a logic change, the job must be redeployed so the updated logic is written to the operating system.

The following script redeploys all of the jobs in a folder. Since the deployed job metadata objects already exist and contain information such as the UNIX deployment directory, the script syntax can be simplified. For example, the FOLDER, DEPLOYMENTDIR, SERVERMACHINE, and SERVERPORT options are no longer required. The following script is an example of the automated redeploy process:

```
# Redeploy
/path/to/sas94/install/SASDataIntegrationStudioServerJARs/4.8/DeployJobs \
-host <metadata_server_host> -port 8561 \
-user sasadm@saspw -password "XXXXXXXXX" \
-deploytype REDEPLOY \
-objects "/Shared Data/Batch Test" \
-sourcedir /path/to/scripts/batch_redeploy/deployed_jobs \
-metarepository Foundation \
-appservername SASApp \
-serverusername batch_id \
-serverpassword "XXXXXXXXX" \
-log /path/to/scripts/batch_redeploy/redeploy.log
```

DYNAMICALLY BUILDING THE REDEPLOY SCRIPT

When supporting a large development team, it may be difficult for the ETL Tech Lead to be aware of every job object to be redeployed. The SINCE option may be a great solution to this problem, but some shops may prefer to explicitly list each job in the redeploy script.

One approach could be to simply maintain a list of jobs to be redeployed in a centralized location using Excel or notepad. Each developer could update the list as his or her tasks are completed. Once the development cycle ends, the Tech Lead could paste the list of jobs in a program that dynamically builds the script. The following SAS code is an example:

```
data work.jobs_to_redeploy;
infile datalines truncover;
input job $char100.;
datalines;
/Shared Data/Batch Test/Project_100_Sample_Job
/Shared Data/Batch Test/Project_110_Sample_Job
/Shared Data/Batch Test/Project_120_Sample_Job
/Shared Data/Batch Test/Project_130_Sample_Job
/Shared Data/Batch Test/Project_140_Sample_Job
/Shared Data/Batch Test/Project_150_Sample_Job
/Shared Data/Batch Test/Project_160_Sample_Job
/Shared Data/Batch Test/Project_170_Sample_Job
/Shared Data/Batch Test/Project_180_Sample_Job
```

```
/Shared Data/Batch Test/Project_190_Sample_Job
/Shared Data/Batch Test/Project_200_Sample_Job
;;;;
run;


data _null_;
file "/path/to/scripts/batch_redeploy/redeploy.sh";
set work.jobs_to_redeploy end=last;

if _n_ =1 then do;

   put '#Dynamic Redeploy';
   put '/path/to/sas94/install/SASDataIntegrationStudioServerJARs/4.8/DeployJobs \';
   put '-host <metadata_server_host> -port 8561 \';
   put '-user sasadm@saspw -password " XXXXXXXXX " \';
   put '-deploytype REDEPLOY \';
   put '-sourcedir /path/to/scripts/batch_redeploy/deployed_jobs \';
   put '-metarepository Foundation \';
   put '-appservername SASApp \';
   put '-serverusername batch_id \';
   put '-serverpassword "XXXXXXXXX \';
   put '-log /path/to/scripts/batch_redeploy/redeploy.log \';
   put '-objects "' job +(-1) '" \';
end;

else if not last then
   put '"' job +(-1) '" \';
else do;
   put '"' job +(-1) '" ';
   put '#End of script';
end;

run;
```

ITEMS TO NOTE

The following information may prove to be helpful when using the DeployJobs utility:

- The passwords listed in the PASSWORD and SERVERPASSWORD options can be encoded with proc pwencode.
- If DeployJobs cannot create deployed job metadata objects in the intended folder, they will be created in the /Shared Data metadata folder.
- The directory specified in the DEPLOYMENTDIR option must be defined as a deployment directory. This can be done with the Schedule Manager pop menu (right mouse button) in SAS Management Console.
- The trailing backwards slash used in these examples allows the UNIX script to be written on multiple lines. Trailing slashes should **NOT** be used within any DeployJobs options.
- The following DeployJobs warnings, if produced, may be ignored. They are Java Garbage Collection output related to releasing memory between multiple deployments or redeployments:

  23:59:24 WARN (fD,r0) A connection was returned to the factory by the garbage collector (fD,c0)
- Even though the focus of this paper is on UNIX and Linux, the command-line deployment tool is also available on Windows.

CONCLUSION

Deploying and redeploying SAS Data Integration Studio jobs is an essential step in the progression from business requirements to the data warehouse that is often the catalyst for solving business problems. By using the revised batch deployment utility, the ETL Tech Lead can create an automated process that is easier and more efficient than manually deploying and redeploying jobs. The goal of this paper is to help SAS Data Integration Studio development teams adopt this utility.

ACKNOWLEDGMENTS

The author would like to thank Tom Hahl of SAS Technical Support for his assistance with this material.

RECOMMENDED READING

- SAS Data Integration Studio 4.902 User's Guide: http://support.sas.com/documentation/cdl/en/etlug/69395/HTML/default/viewer.htm#p1jxhqhaz10gj2n1pyr0hbzozv2f.htm
- Problem Note 57103: Performing command-line batch deployments in SAS® Data Integration Studio generates code that contains USER= and PASSWORD= instead of AUTHDOMAIN

CONTACT INFORMATION

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

Jeff Dyson
The Financial Risk Group
jeff.dyson@frgrisk.com
frgrisk.com