Paper 419-2013

Making do with less: Emulating Dev/Test/Prod and Creating User Playpens in SAS® Data Integration Studio and SAS® Enterprise Guide®

David Kratz, d-Wise Technologies

ABSTRACT

Have you ever required a Dev / Test / Prod environment but found yourself, for whatever reason, unable to lay down another SAS Installation? Have you ever discovered that your results have been overwritten by a team member? Our ability to use SAS is shaped by the environment in which the software is installed, but we often don't have as much control over that environment as we'd like. However, we can often emulate the setup we'd prefer by configuring the one we have. This paper explores this concept using techniques which can be applied to development in SAS Data Integration Studio and SAS Enterprise Guide.

INTRODUCTION

While building a data warehouse for a client, an interesting situation arose: our development installation of SAS was reallocated to another department. The end result was that we had to constrain our development efforts to a single machine. This, combined with software changes made in SAS 9.2 and beyond, soon resulted in a secondary problem. The data warehouse was built as a result of a flow of jobs, most of which depended on data provided by previous jobs. With multiple developers coding and debugging simultaneously, it was not uncommon for one person's test source data to be overwritten by someone else's test output.

Under normal circumstances, we would have resolved these issues by laying down another instance of SAS. For various reasons that was not possible, and our only option was to either soldier on or get creative. We chose the latter, and the solutions we found follow. More generally these techniques should be applicable in any single machine environment in which there are multiple users.

The situation did bring something into sharp relief. How well we can utilize software is often influenced by the environment in which we use that software. Unfortunately, even at the Admin level, that environment can still be beyond our control thanks to external factors. However, the flexibility of SAS combined with a little lateral thinking can sometimes allow us to emulate the environment we wish we had. While the solutions presented here are specific, the general mindset should be applicable in a wide range of situations.

USER PLAYPENS

WHAT ARE USER PLAYPENS?

User playpen is a term we used to reference a space (in this case a set of libraries) where users could run jobs without fear of overwriting each other's data. A further tweak allows them to access and modify shared data such as reference tables without permanently overwriting the original file. Much like a child's playpen, the space allows users to do what they want without fear of harming themselves or others. This technique will be most useful in systems in which users are running processes which feed into each other (the output of one job is the input of another), or users find that they are often simultaneously running a job which writes to a non-work library.

PHYSICAL INSTALL SIMULATED

From a user stand point, when this technique is in place, it will be as if they have the play-penned libraries to themselves. However, they will of course still be sharing system resources when actually running jobs.

HOW TO ACHIEVE THIS EFFECT

The key technique to our implementation of this effect is dynamic library reassignment through the use of pre-assigned libraries. Put more simply, the location a library points to is changed based on some set of factors. While this would be easy to do in regular SAS code using SAS Macro, it is more complicated to get such changes to work correctly in applications like SAS Enterprise Guide or SAS Data Integration Studio.

Modify autoexec.sas to pre-assign the play penned libraries

Modify the SASapp appserver_autoexec_usermods.sas to contain something like the code sample below for each play penned library. This file is usually located under your SAS configuration folder on the application server in a subfolder like: …/lev1/SASapp. The libraries must point to extant folders on the application server file system, which

might entail creating new folders. Additionally, confirm that there is adequate free disc space for the work that is expected to be done in the playpens.

Autoexec.sas is a SAS program which is run at the beginning of a SAS session. In base SAS, it is run as the application starts. In SAS Enterprise Guide and SAS Data Integration Studio, the application server's autoexec.sas is run whenever almost anything is done, from opening a data set to running code. Anything that can be done in normal SAS code can be done in the autoexec, including library assignment.

```
%macro dynalib;
    %if &SYSUSERID = dkratz %then %do;
          libname example "c:\example\users\dkratz ";
    %end;
    %else %do;
          libname example "c:\example\users\public ";
    %end;
%mend dynalib;

%dynalib;
```

&sysuserid is a SAS automatic macro variable, which contains the user id or login of the owner of the current SAS process. In essence, this code sample assigns the example library based on the user running the process.

For the purposes of this example, only the SASapp autoexec is modified. However, that is not the only autoexec which could be changed. Many components of an application server have their own autoexec files, which inherit the autoexec settings of their parent application server. The net result of this is that the processes which use the dynamically assigned libraries can be controlled by which autoexec receives the changes. A change made in the autoexec at SASApp would affect all sessions using that application server. A change made in the autoexec of SASapp's workspace server would be significantly more confined.

Set the play penned libraries to be pre-assigned

1. In SAS Management Console, select the Data Library Manager plug in and expand the libraries folder.
2. Select the first play penned library and right click. Select properties.
3. In the window that appears, select options and then click on the advanced options button.
4. In the window that appears, check the "Library is pre-assigned" box. For Pre-assignment type, choose "By external configuration."
5. Ensure that the library's libname matches one specified in autoexec.sas.
6. Repeat steps 2 – 5 for each play penned library.

Under normal circumstances, the assignments made in autoexec.sas would be overwritten by the metadata library assignments used in SAS Data Integration Studio and SAS Enterprise Guide. However, if a library is set in metadata to be pre-assigned, the programs do not assign their own value, and the autoexec supplied value remains.

This and the preceding steps are the only ones required for the dynamic reassignment. The remaining steps describe optional tweaks that can be made to the autoexec code to provide additional functionality.

Optional: Concatenating a read only library

1. Create a libname statement pointing to an arbitrary folder.
2. Create a libname statement with the option access=readonly which points to some collection of data sets.
7. Create a libname statement which concatenates the two previous libnames, with the read only libname coming last.

Modify the earlier autoexec example following these steps, and it becomes something like this:

```
%macro dynalib;
    libname omega "c:\example\reference" access=readonly;
    %if &SYSUSERID = dkratz %then %do;
          libname alpha "c:\example\users\dkratz";
    %end;
    %else %do;
```

```
            libname alpha "c:\example\users\public";
        %end;
        libname example (alpha omega);
    %mend dynalib;

    %dynalib;
```

In the code above, the example library will report as containing both the contents of the alpha and omega libraries. Because library concatenations look for files beginning with the leftmost library in the concatenation, when two tables exist with the same name in alpha and omega, only the alpha version will be present in the example library. Similarly, when the example library is written to, tables will end up in the alpha library.

Assume that the omega library contains reference data that multiple users would refer to, and that the alpha library is a user's empty private folder. When the user examines the example library, it will report as containing the reference data. They can run whatever DATA step code they like on the reference tables, and what they will create is a modified copy actually located in their own folder. To revert to the original data, a user simply has to delete the table in the example library, which removes the copy from the alpha library, allowing the original in the omega library to be visible. Since the omega library is readonly, any attempt to update in place / delete a table which does not have a duplicate in alpha from it (the omega library), will result in an error.

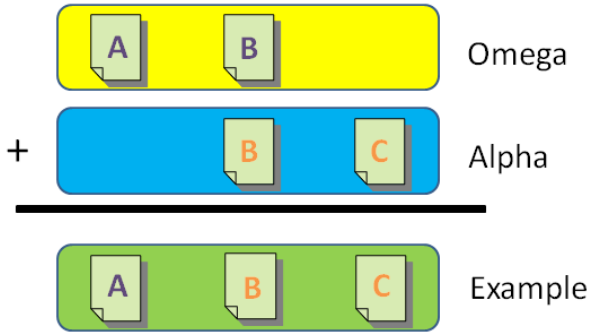

Figure 1. Visual Explanation of Library Concatenation

Optional: Allowing underlying physical structures to determine access

1. Create a folder structure in which all authorized users have a folder that contains a copy of the play penned library folder structures. Create a structure for a public user as well.

3. Modify the permissions of this folder structure such that each authorized user has access only to their folder.

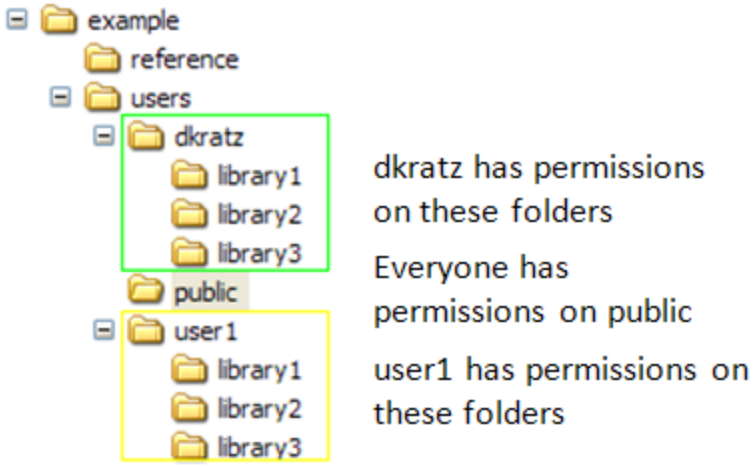

Figure 2. Example of physical folder structure and permissions for user folders

8. Use SAS Macro coding to test for the existence of a user's folder before assigning libraries. If the folder exists, the user is assigned their own folder. If the folder doesn't exist, they receive libraries pointed only at the public user folder, with no concatenation.

Modify the earlier autoexec examples following these steps, and it becomes something like this:

```
%macro dynalib;
    libname omega "c:\example\reference" access=readonly;

    %if %sysfunc(fileexist("c:\example\users\&SYSUSERID")) %then %do;
          libname alpha "c:\example\users\&SYSUSERID";
    %end;
    %else %do;
          libname alpha "c:\example\users\public";
    %end;
    libname example (alpha omega);
%mend dynalib;

%dynalib;
```

While it's true that autoexec code can do anything that regular SAS code can do, it's important to recall that running this code takes time. In the case of the Appserver autoexec, it's code that will be run very often. Thus the longer it takes for the macro code to determine which library assignment a user is given, the less responsive SAS Data Integration Studio or SAS Enterprise Guide will seem.

One relatively quick method is to have the autoexec code check for the existence of a physical folder or file named after the user. The variable value to test for, the sysuserid automatic macro variable, is controlled by the existing SAS login process. Since administrators can lock down a directory structure such that only they can modify it, they retain control of who will be granted access using the redirected libraries and who won't.

Consequences of this configuration

There are potential security concerns for the configuration we have just detailed. Since autoexec.sas now controls how certain libraries are assigned, it should be locked down such that only administrators can edit it. Similarly, the per user file structures should have their permissions secured such that only the owners and administrators can read

or edit as appropriate.

Another possible consequence is user confusion. Users will need to be educated about the workings of the concatenated read only libraries. For example, a user might believe that they are editing the shared source data, only to have those changes applied to their play penned data instead. Alternatively, they might be confused when they drop one of their play penned data sets, only to have it immediately replaced by the shared data set of the same name.

Administrators are not immune from this confusion. Investing some of the SMC's normal authority (in this case library assignment and user access) into autoexec.sas creates a more opaque configuration and hidden places for things to go wrong. It can also require additional configuration steps to be taken when editing users. All of this requires additional training and documentation for the administrators. Simply put, there might be a bit of a learning curve.

EMULATING DEV / TEST / PROD ENVIRONMENTS

WHAT ARE DEV / TEST / PROD?

Dev, Test, and Prod refer to the three most common phases of software development: development, testing, and production. Ideally, these phases take place in their own environments, which usually inherit the names of their respective phases. Dev is where all coding takes place, and the environment often features only a small subset of the total data. Test is as close a mirror of the production environment as possible, and the location where software is tested. Prod is the production environment, where finished code is accessed by users. In a perfect world, code would never be edited in Prod, all changes should flow from Dev through Test, and then simply be pushed to Prod. This separation allows for the impact of any changes made to be isolated and assessed before it can affect the end users' experience.

PHYSICAL INSTALL SIMULATED

Dev / Test / Prod emulation simulates three separate physical installations of SAS on a single machine. A properly configured user will be able to interact with only those tables or jobs that are appropriate for their assigned environment. Specially designated users will be able to promote between environments. Note that this technique differs between SAS Data Integration Studio and SAS Enterprise Guide because of the different ways they store jobs.

HOW TO ACHIEVE THIS EFFECT IN SAS DATA INTEGRATION STUDIO

Physical files

Allocate spaces to hold the physical data (tables, indexes, etc.) for each environment to be emulated. If possible it is a best practice to place them on separate directly mounted drives. This reduces the chances that a single drive failure will disturb multiple environments.

Custom metadata repositories

By default, SAS stores all of its metadata in the foundation repository. However, it is possible to create your own repositories, which are known as custom repositories. Each of these stores metadata in a different physical location, but they are all represented as a top level metadata folder in the metadata folder structure. As with the underlying physical files, whenever possible store these repositories on distinct drives.

1. Create custom metadata repositories for each environment. Name them appropriately; for example, Dev, Test, Prod.

1. Create a folder in lowest production level repository with a name that is not specific to any given environment. For example, Root would be preferable to Dev or Dev_root. This will make the promotion processes discussed later easier.

Manage metadata permissions to restrict access between environments

1. Assign each user to a group with appropriate permissions to read and edit the metadata and files of one of the repositories.
2. Designate new or existing accounts as promotion accounts. These accounts should be added to a second group, such that they have privileges both on the group they promote from and the group they promote to. For example, the TEST promotion account would have privileges to both Dev and Test, etc.

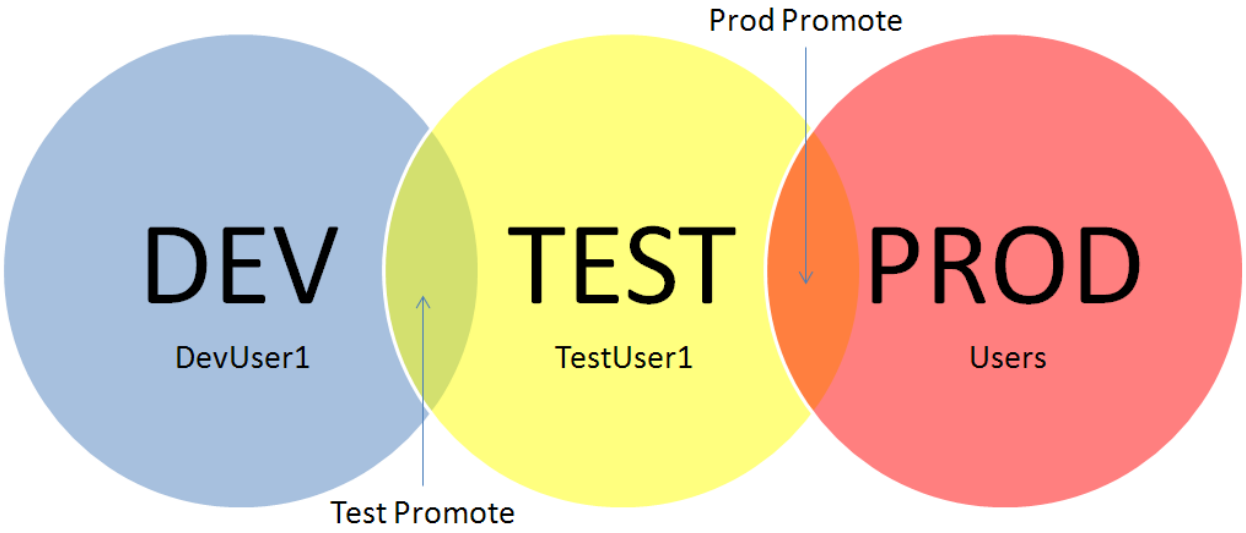

Figure 3. Example user groupings for the purposes of assigning physical and metadata permissions in Dev / Test / Prod Emulation

Manage physical permissions to restrict access between environments

To prevent possible mishaps, edit the user permissions on the underlying file stores to be in line with the metadata permissions. That is to say, development users should not have read or edit privileges on test or production data, and so on. Promotion accounts should, as with metadata, have access to both the environment they promote from and the environment they promote to.

Physical promotions

Using the appropriate promotion user account, perform a physical copy of the data from one repository's physical file store to another, newly modified files overwriting their equivalent file on the new environment. While not technically necessary, it will make things a lot easier to manage if the structure of the file stores (the names and locations of folders within the store) remains consistent across all environments.

The first metadata promotion

1. Copy the root level folder of the source repository and paste it into the target repository. Click OK if prompted about a warning occurring during the copy process. Note that this should not be done piecemeal. If the initial promotion contains enough metadata that copying and pasting all of it causes an error, then the transfer will need to be done using the import/export method (described below) in smaller chunks.

1. Rename all libraries that were copied over. Library names must be unique in metadata, so the copies will have been given names of the type "Copy of x", where x was the name in the previous repository. The new name should reflect the new environment; for example, "Copy of x" might become "x_TEST".

3. Change the underlying physical location of the newly promoted libraries. Newly promoted libraries will still be pointing to their previous location. Change the location of each library to point to its equivalent location in the new development environment.

The first metadata promotion differs from all following metadata promotions. In SAS Data Integration Studio, if a folder structure that contains a job and all of its component parts is copied to a custom repository, it will automatically reassign the job's components to the newly copied ones. However, it does not overwrite existing metadata objects, so it is not practical for periodic promotion.

Metadata promotions

1. Export data from the source repository. Right click on the source repository's root level folder and choose export -> SAS package. Use the "Clear All" button to clear all selections, and then select only those items that you wish to move over. If transferring jobs, it is vital to include all dependencies. Complete the export as per normal.
4. Import data into the target repository. Right click on the target repository or its root level folder, and choose import -> SAS package. Select the package that was chosen in the preceding step. If any tables with an assigned library were included in the import, the option to select a library will present itself. Make sure to select the libraries which correspond to the new environment.

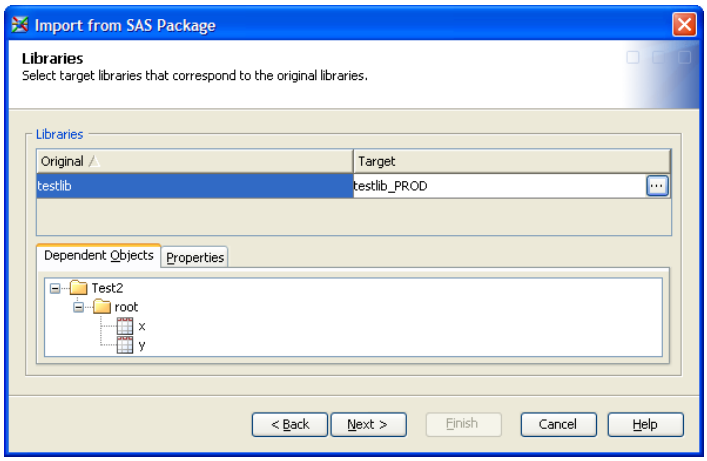

Figure 4. Example of the library selection screen encountered when importing a job which contains tables

All promotions after the first should be done by exporting metadata from the source repository and importing it into the target repository. Unfortunately, not all items can be transferred in this way. Libraries and custom transformations are among the exceptions. Libraries may be copied and pasted or exported / imported as described above only during the libraries first promotion. All other promotions will require that changes are made manually to each environment's copy of the library. Doing otherwise will simply create an additional copy of the library, which won't be referenced by any table or job.

Custom transformations behave similarly, except they can never be promoted. The same transform is referenced across all levels of development. There is no way to replace one in a promoted job short of manually deleting the previous transformation and replacing it with the updated version. Consequently, custom transformation should be stored outside the emulated repositories, and extreme care should be taken when editing them.

How to achieve this effect in SAS Enterprise Guide With the SAS Intelligence Platform

SAS Enterprise guide is a dynamic application whose features and usage can vary widely depending on the other SAS products installed on the machine. Achieving this effect on a machine that is running the SAS Intelligence Platform (SAS Data Integration or SAS Business Intelligence have been installed) is very similar to the SAS Data Integration Studio guide provided previously, with the exception that Enterprise Guide has physical project files, rather than metadata jobs. This has the following consequences:

- Enterprise Guide Project files should be stored in the same permissions managed space as the underlying data
- Promotion of jobs now happens during the physical promotion step, rather the metadata promotion step.

How to achieve this effect in SAS Enterprise Guide Without the SAS Intelligence Platform

There are multiple ways to achieve this effect in SAS Enterprise Guide without the SAS Intelligence Platform, each

with varying implementations and drawbacks. The best choice will depend on the specifics of the environment in which the technique is being used. For sake of contrast and illustration, the following implementation avoids using metadata in favor of dynamically assigned libraries.

Physical files

Allocate space to hold the physical data (tables, indexes, etc.) and program files for each environment to be emulated. If possible it is a best practice to place them on separate directly mounted drives. This reduces the chances that a single drive failure will disturb multiple environments.

Manage physical permissions to restrict access between environments

1. Assign each user to a group with appropriate permissions to read and edit the physical files of one of the repositories.

5. Designate new or existing accounts as promotion accounts. These accounts should be added to a second group, such that they have privileges both on the group they promote from and the group they promote to. For example, the TEST promotion account would have privileges to both Dev and Test, etc. These accounts should not use Enterprise guide, as the suggested autoexec code would default them to only one of their granted environments.

Setup dynamically assigned libraries to control access to the environments in SAS Enterprise Guide

Use the steps described in the first half of this paper to setup dynamically assigned libraries which point to the various environment file shares. Adjust the autoexec code to assign users to their appropriate repositories. For example, the previously suggested method of using folder presence would work.

Promotions

Using the appropriate promotion user account, perform a physical copy of the data and jobs from one repository's physical file store to another, newly modified files overwriting their equivalent file for the new environment. While not technically necessary, it will make things a lot easier to manage if the structure of the file stores (the names and locations of folders within the store) remains consistent across all environments.

CONSEQUENCES OF THIS CONFIGURATION

This setup is useful from an organizational / usability standpoint, but it doesn't provide all of the advantages of having actual distinct machines for each environment. Specifically it creates a single point of failure for three separate and distinct groups of users, and in doing so limits the kind of changes that can be easily tested across production levels.

Say for example that a development user accidentally runs a query that creates a Cartesian product. Depending on your configuration, your production users could be impacted. Worse, say that the development user manages to crash the metadata server. Production has now been crashed as well. In addition, if there is only one machine, then it is impossible to test hotfixes or upgrades before applying them to the production environment. This is a significant challenge, and one of the most important reasons that use of multiple machines is the optimal solution.

To provide some real world context, the company for which this method was originally developed is based in Manhattan. Their building was flooded during Hurricane Sandy, knocking their SAS server offline. While they had physical backups of their data, they had no available backup servers on which to run it. A single event over which they had no control didn't just stop development, it also knocked production offline.

Finally it's worth noting that at least in SAS Data Integration Studio's case, using this technique results in a multiplication of the amount of metadata created by any process which makes its way to production. From a usability standpoint, this might impact the speed at which jobs and tables load from the metadata server. Additionally, metadata corruption, while rare, is a very real threat when using the SAS metadata server. The problem is exacerbated as the amount of metadata grows. The specifics of how much metadata is too much will vary per system, but if a system is already experiencing slowdown or corruption issues, the application of this technique might not be advisable.

CONCLUSION

In conclusion, the SAS environments which we use are often shaped by external factors beyond our control. However, it is often possible through configuration to simulate the environment we would prefer. In this paper we have discussed two specific methods, Dev / Test / Prod Emulation and user playpens, as examples of what is possible through configuration. Respectively, these allow a single machine to simulate multiple development environments and for users to operate as if they were the only ones interacting with a set of libraries. However, these

techniques are not without their downsides. The first fails to capture all the benefits of separate development machines and the multiplication of metadata it creates can itself lead to problems. The second can create additional security concerns and might be confusing to uninitiated users or administrators.

CONTACT INFORMATION

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

Name: David Kratz
Enterprise: d-Wise Technologies
E-mail: David.Kratz@d-wise.com
Web: http://www.d-wise.com/