PhUSE 2007

Paper DM02

Integrating Clinical Data Transformations using DI Studio and SAS Drug Development

Alistair Dootson, Dootsonic Ltd, Aylesbury, UK

ABSTRACT

DI Studio is a SAS tool that lets the user visually define executable processes, which until now had to be created and deployed using traditional SAS programming techniques such as using the program editor window within Foundation SAS.

Part of the statistical programming role involves creating a derived database allowing easier production of statistical tables and listings. Pharmaceutical Company Data comes from many sources such as CRO or EDC systems. In this example, the data is loaded into SDD then transformed into a format that can be reported on.

This paper will examine how the data gets from its source to its target whilst maintaining a validated state, the target options and what happens once the metadata transformation is complete and to demonstrate how SAS DI Studio enables users to produce derived data, the benefits the tool brings to the business and the integration with other environments such as SAS Drug Development and BI.

INTRODUCTION

The purpose of the paper is to examine several aspects of managing clinical data in a regulated environment. A lot of focus is trained on data management, where the data is stored during the data collection and cleansing elements of a trial. But once the data leaves the hands of data management, the regulatory focus tends to change. But how can the study database be traced through to deliverables? How can changes made be documented? How can another Statistician or Programmer maintain another's code? The paper will look to cover these elements.

This paper will in the main focus on DI, and then move to show how DI can interact with other SAS tools including SDD and BI. There are several sections to cover: Metadata, Study Database (Raw Data), Derived Database, SAS Data Integration Studio, SAS Business Intelligence, SAS Drug Development. These will be covered in turn and brought together by a process at the conclusion.

METADATA

Metadata is basically information about data sources. It is data about data that describes the make up of the tables including column lengths, type, labels and any formats associated. In our example, we find metadata datasets to be an empty shell of a dataset (no observations, but several variables) which are used to define columns in tables in DI and BI tools.

STUDY DATABASE

A Study Database can be defined as a deliverable from Data Management that is the locked clinical trial database. The Statistical department can take the locked database and use as a basis for performing statistical analysis.

DERIVED DATABASE

A Derived Database is the Study Database manipulated with additions of derived variables and processed in such a way that the data can be reported upon. Such derivations include adding treatment information and unblinding of study data plus any other variables or transformations required to shape the data ready for clinical trial reporting. It can be described as being "One proc away" from the tables, listings and graphs.

SAS DATA INTEGRATION STUDIO

SAS DI Studio is, as it suggests, an application that allows a flexible way of integrating many different forms of data. Benefits of DI Studio include the ability to define standard lists of targets (definitions of tables, standards, for example target CDISC format datasets), easily define input tables from SAS, Oracle, EXCEL formats amongst others and display a useful information map of the process taken to define the target data. DI Studio also provides graphical illustrations that are easy to follow and significantly is a code generation tool. The most important feature is the Metadata that acts as the glue between the input and output data. It allows traceability and documentation of a process that may easily get lost in an abundance of programs.

SAS DRUG DEVELOPMENT

SDD acts as a central repository for any files, as a data warehouse or for execution of SAS programs in a regulated environment. It can be hosted remotely at SAS or on-site, however having a hosted application can give great benefits regarding validation, hardware and maintenance costs. It can improve the speed of implementation, IQ/OQ testing and be easier to develop installation requirements. The application is web based giving remote, global flexibility and has features such as version control and Electronic Signatures to enhance its regulatory package.

SAS BUSINESS INTELLIGENCE

SAS BI can act as a way to run stored processes created as part of the entire process, these are effectively SAS programs wrapped up in a macro and can be accessed by any users with access, BI can also be used as a means to create the deliverables at the culmination of the process, in our example to develop SAS code for Tables, Figures and Listings. SAS BI acts as a reporting feature using Add ins to Microsoft Office which can run Stored Processes to display output or using Web Report Studio, a web based tool to create reports with, or even as a development tool for creating output. Tools such as Enterprise Guide can be used to create quick QC outputs, or even to develop code in as an alternative to Foundation SAS.

ARCHITECTURE

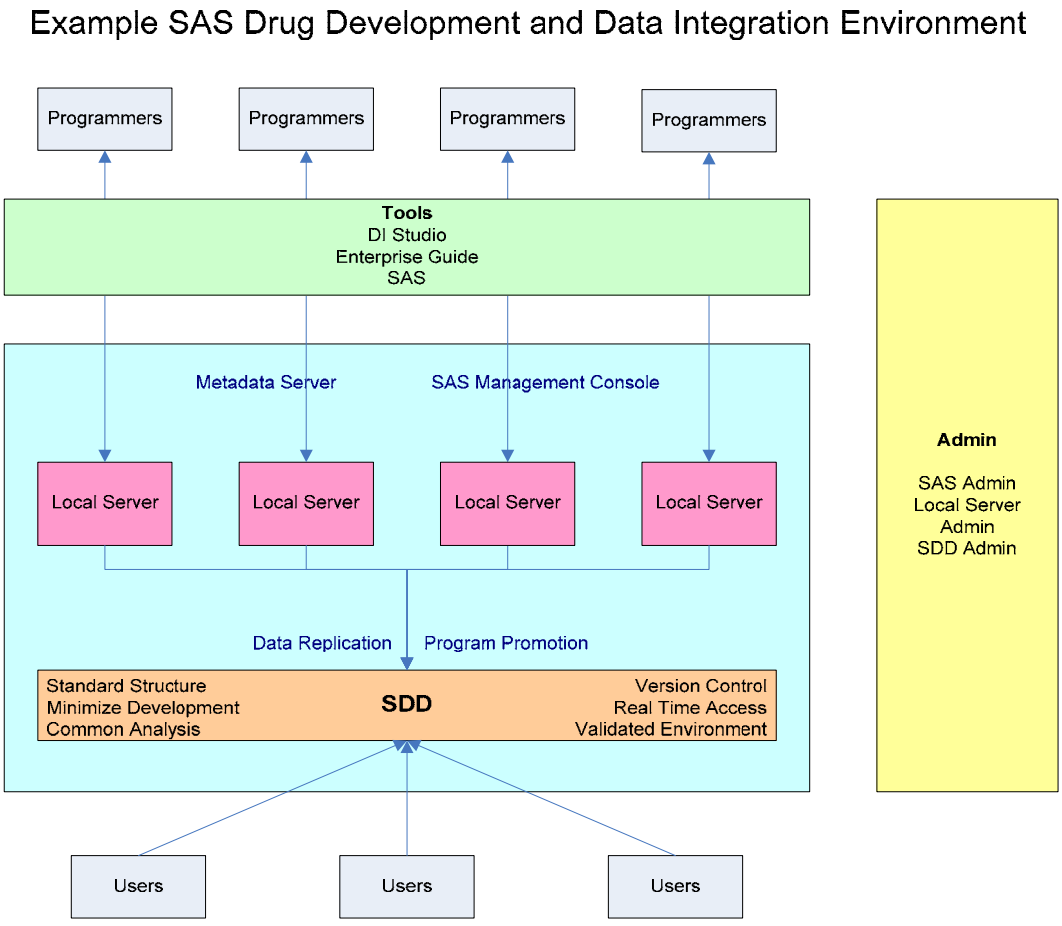


Figure 1 above demonstrates the overall architecture of the environments. Programmers work locally and using promotion and replication interact with the SDD environment. Users can log on to SDD to view data and output without needing the local environment. Administrators oversee the environments.

THE PROCESS

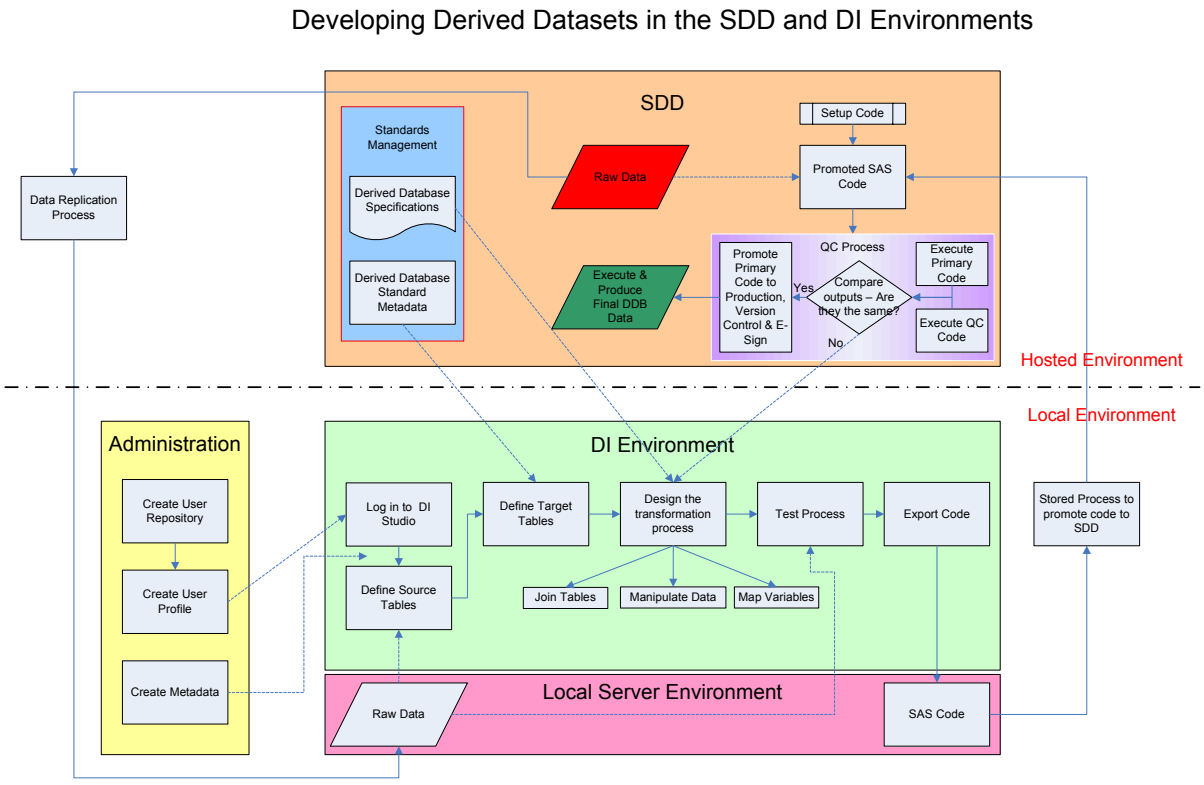


Figure 2 above demonstrates the overall flow between SDD and DI environments. In short, the flow starts with Raw data in SDD which is replicated to the local server environment. DI Studio is used to create the code for the transformations using a graphical interface, following specifications and referencing standard metadata that is the basis for target tables. The final code is exported to the local server before being promoted to SDD via another Stored Process. The code is then executed in the SDD environment which populates the Derived Database.

STEP 1 – SDD

SAS Drug Development acts as a main repository for any data, programs and even documentation to do with clinical trials. In the following example, once a Data Management department have locked the database and supplied it to the Statistics department in a Standard format it is stored in the SDD environment. Version Control is switched on and it can be electronically signed. Should any changes need to be made, the version of the data is incremented each time it is overwritten, but versioning cannot be switched off once it is enabled so it pays to be certain about the data before uploading the final version to SDD.

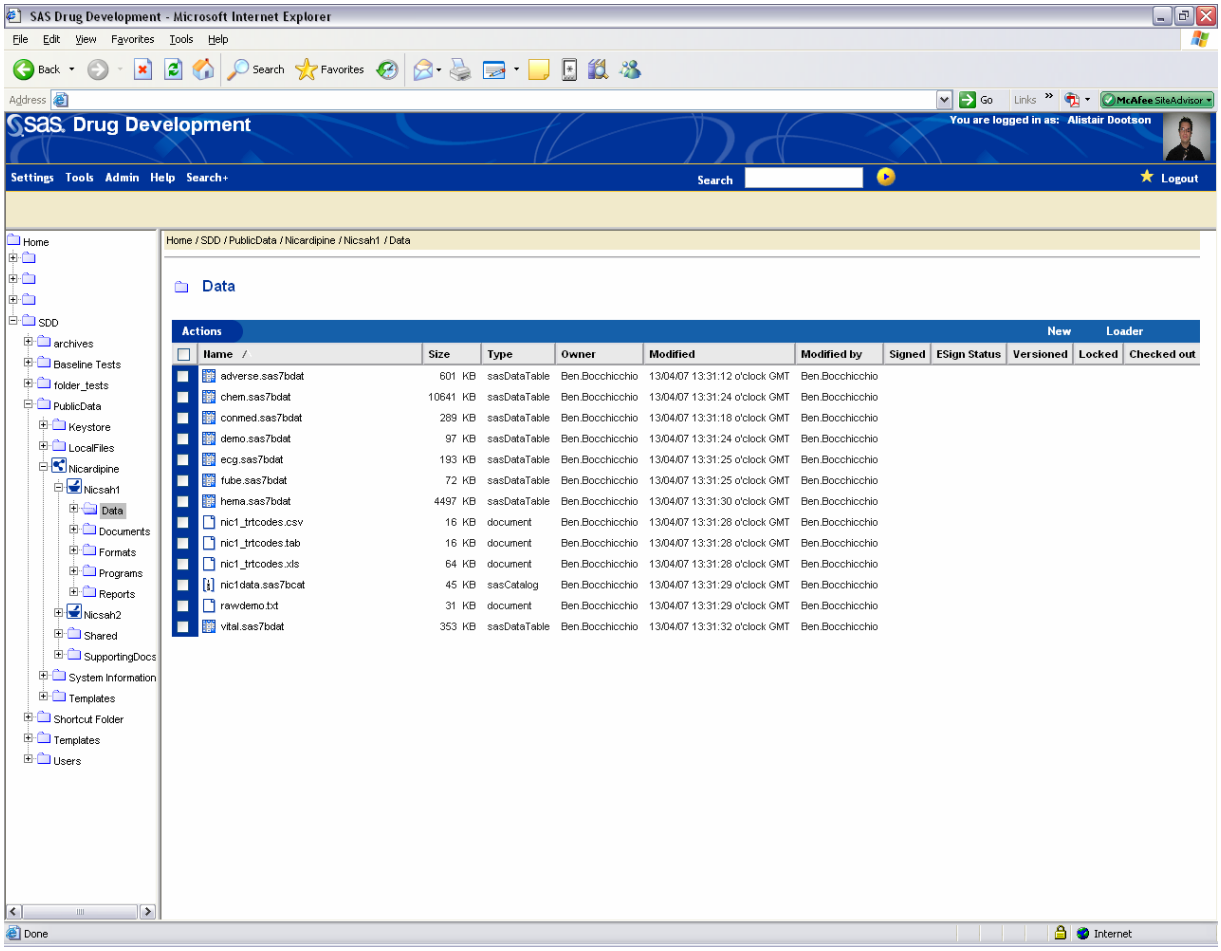


Figure 3 An example data repository in SAS Drug Development

Data and Format library standards that a company might have should be stored and version controlled in SDD. This includes any general documentation, but also metadata datasets. That is, an empty shell of the standard dataset structure needs to be stored so that it can be used to define transformations later on. There should be metadata for Standard Database structure and also Derived Database structure. In DI Studio, the Standard structure can be used to check the DM deliverable, but the derived database structure is the goal of this example process. Any specifications for creating the Derived Database should also be stored in SDD in the same way.

In order for the data to be used in the DI environment, a stored process should be created to replicate any data down to the local servers.

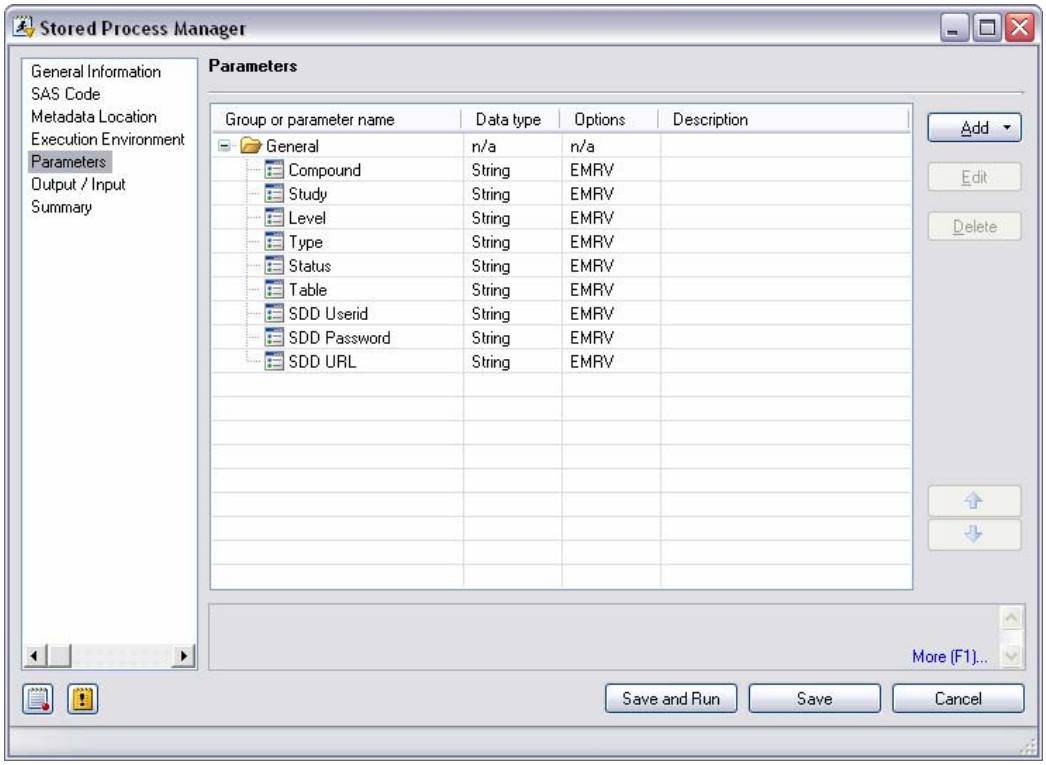


Figure 4 – A Stored Process to replicate data from SDD to The Local Server showing the screen where Parameters are defined. The Stored Process defines the origin and the destination of the files to be replicated, the user can specify a type of file to move such as datasets. The idea behind the replication is for the data to be stored in the repository, but to be available to use in the local tools to produce the necessary programs.

With the data being stored centrally, any region that has a local server can download it to work locally. This enables global working, but it must be remembered that SDD is the central repository that is replicated from, and not written to in this process.

STEP 2 – ADMINISTRATION

There are several administrative tasks that need to be completed before performing the data transformations in the DI environment.

Create Repositories – In the Management Console, there should be a hierarchy of repositories set up for each study to reside in. For example, the top level repository is always Foundation, which holds user information and general metadata information. Next level down is a Standards repository which holds the global standards for a company. At the next custom level would be a Client or Compound level repository and beneath that a custom Study level repository. Finally, beneath each Study would be several project repositories representing work areas for the programmers involved.

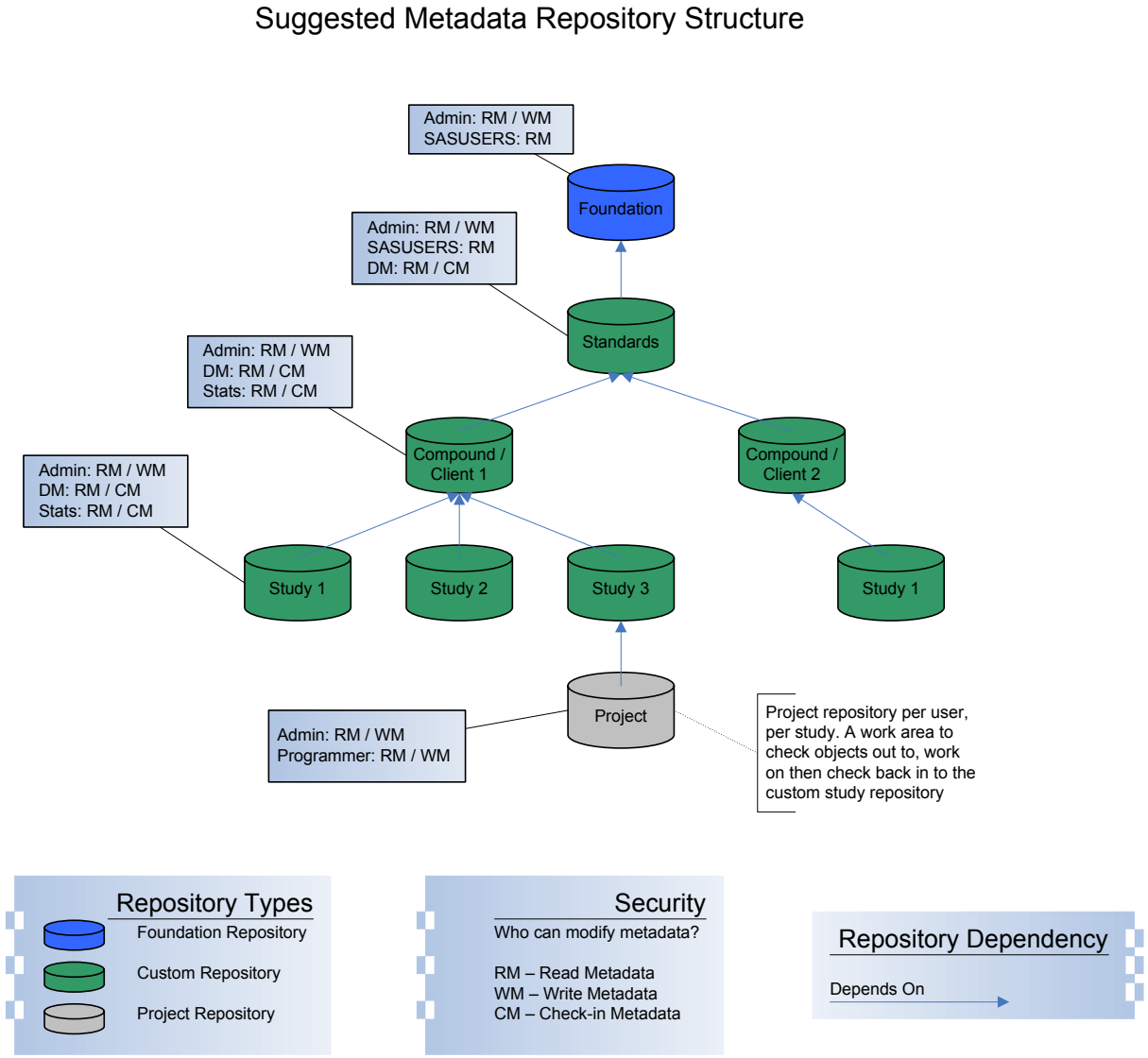


Figure 5 An example Metadata repository in SAS

Create User profile – each user requires a profile that matches to the repository they are working in. The profile can be set up when a user connects to DI Studio, or an administrative function can be to supply the profile to the user.

The administrator executes the stored process entering only the profile name to be created. The SAS code then creates a Metadata Profile that is stored on the user's local machine as an .swa file stored in their Workspaces area. DI Studio is then able to pick up the profile. As part of the stored process, the .swa file can be emailed to the user.

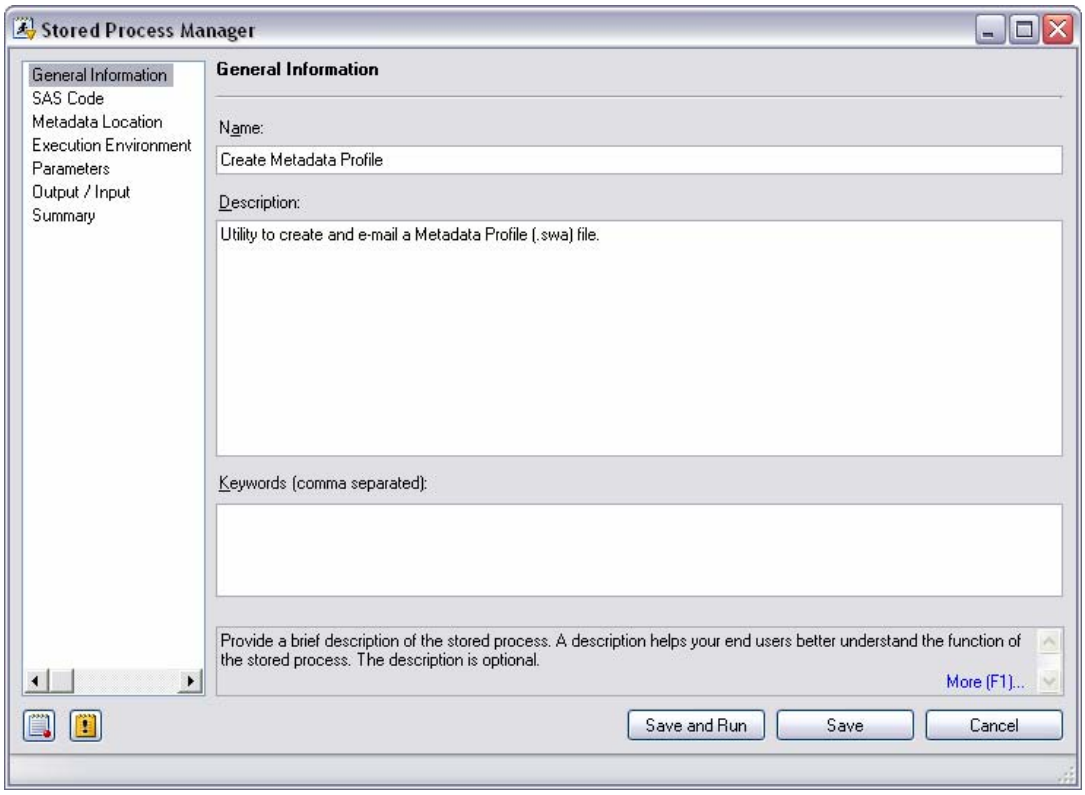


Figure 6 An example of the Stored Process screen to create a User Metadata Profile

Create Metadata – for each study to be standard across the environment, standard metadata should be created for each study. The standard metadata includes data libraries and references. These are held in the study repositories and should be the same for each study to promote standard libnames and references. The programmers would then become familiar with the references required to access data across studies. Other metadata to set up are security features on the libraries such as granting or denying access to users or groups.

By creating the metadata automatically, the administrator can enable standard libraries to be set up consistently for each study. These can be handled by a central csv file that can be read in by the code. The general activities the program performs can be done manually in the SAS Management Console, however to assign 40 libraries and adjust their permissions would be a very laborious task, especially over several studies. Hence the code to reduce time spent.

The program uses the SAS Open Metadata Architecture coding to manipulate the metadata. As can be seen in the code snippets the program creates a new object then sets attributes and associations to it so that it can be referenced by other metadata and read correctly in SAS. Permissions can then be set on the metadata restricting the users as necessary.

```
if foundName ne treeName then
do; /* Create a new tree node */
   put "NOTE: Creating tree node: " treeName;
   rc=metadata_newobj("Tree" ,t_uri , treeName );
   rc=metadata_setattr(t_uri , 'TreeType' , 'BIP Folder' );
   rc=metadata_setassn(t_uri,
                       "ParentTree",
                       "Append" ,
                       Parent_Id);
   rc=metadata_setassn(t_uri,
                       "Groups" ,
                       "Append" ,
                       "&Group_Id");
            Parent_Id = t_uri;

/* Purge the cache to ensure proper data is used */
   rc = metadata_purge();
end;
```

```
.....
rc=metadata_newobj("Directory", d_uri, "Base Path" );
rc=metadata_setattr(d_uri, 'DirectoryName', path );
rc=metadata_setassn(d_uri,
                    "DeployedComponents",
                    "Append",
                    "&webdavServer_Id");
.....
```

STEP 3 – DI ENVIRONMENT

The main body of the data transformation process takes place in DI. Once the user has their profile and working repository set up they can log in to the study in Data Integration Studio (DI Studio). In the simplest form, it is a three step process – Source, Target & Transformation.

Source. User defines the source tables that will be used to populate the derived data. This is done by utilizing the source designer that walks the user through several steps to locate the data and define its location. It is then imported into the repository and available for use in the transformation. It is important to note that although metadata only is required for the transformations, the actual data will be used for testing the process, so a location that contains the actual data should be selected (as the physical location of the data is required).

Target. User defines the target table that will be populated at the culmination of the process. The target designer walks the user through the steps of where to place the final data and pulls in the metadata required to form the target dataset. It is useful to pull this from a set of standards – for example CDISC – stored on the version controlled SDD environment. The beauty of having a standard derived database structure held in a metadata format is that no action is required at this time except to point to that structure.

Transformation. The transformation section is the most flexible area as the specifications will change from study to study and for each dataset, although with a standard database and derived data, the transformations will hopefully become reusable in time. The DDB specifications should be stored in SDD where they are in a validated state and easily accessible globally. The programmer should take the specifications and begin the transformation process. For derived datasets there will usually be a join of one or more tables including randomization information – whether dummy or not. There will also usually be some manipulation and mapping of variables to get them in a state ready for the derived data.

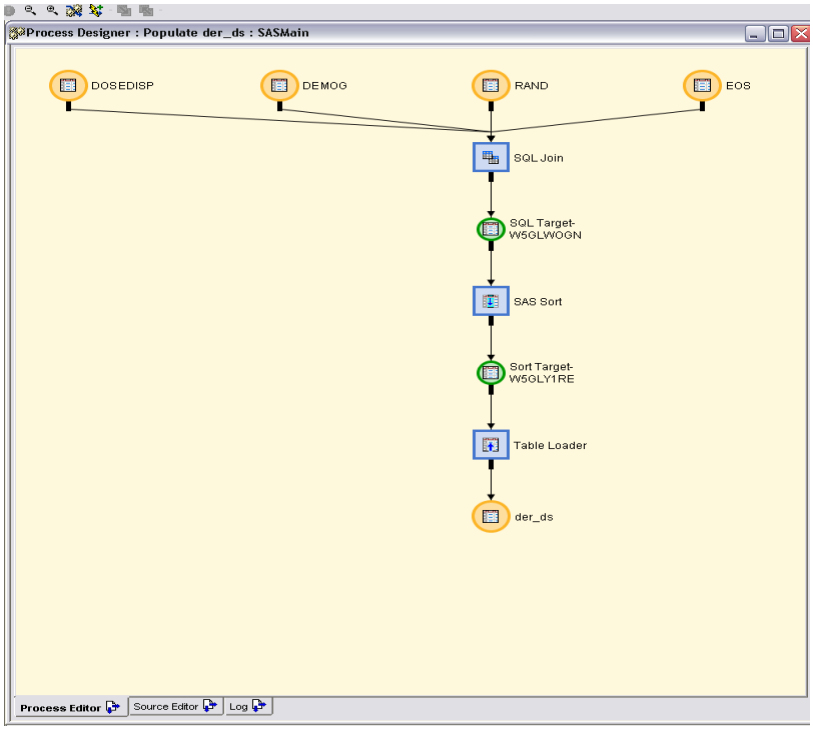


Figure 7 An example Process overview in DI Studio

Join. The join is created using a SQL Join that is well displayed in the DI process editor. All the functions of a PROC SQL join are available including manipulating the join itself (Left, inner etc), selecting variables required, adding where clauses, ordering by etc.

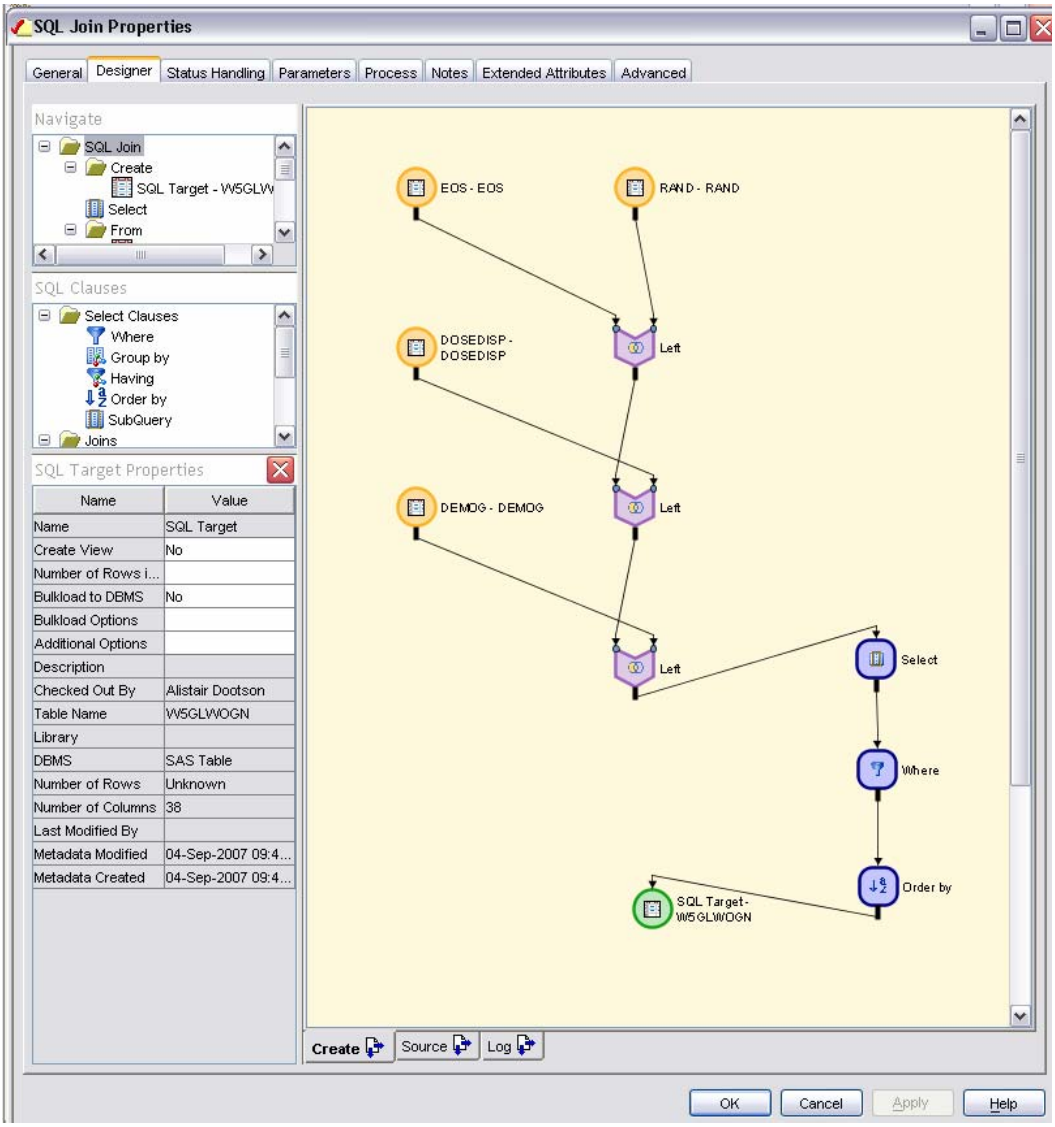


Figure 8 An example Join in DI Studio

Within the join in DI Studio there are the options to perform all actions associated with a proc SQL procedure. The join types can be edited from Inner to Left, Right etc and the join rules specified. Variables can be selected, mapped and renamed if necessary, a Where clause can be added and the output data sorted using the order by statement. All these options can be selected graphically without the need for adding specific code. The final target can be defined as a view or table and is passed back to the overall process.

Manipulate. The variables can be manipulated by using one of the available transformations supplied by DI Studio or by inserting user written code. The traceability of the variables is better using the supplied transformations. A good example is for change in baseline, this can be achieved by splitting the dataset in question and restricting by a baseline visit in one side of the split, yet keeping all rows in the other side before using a SQL Join to merge them back together for mapping to the final variables.

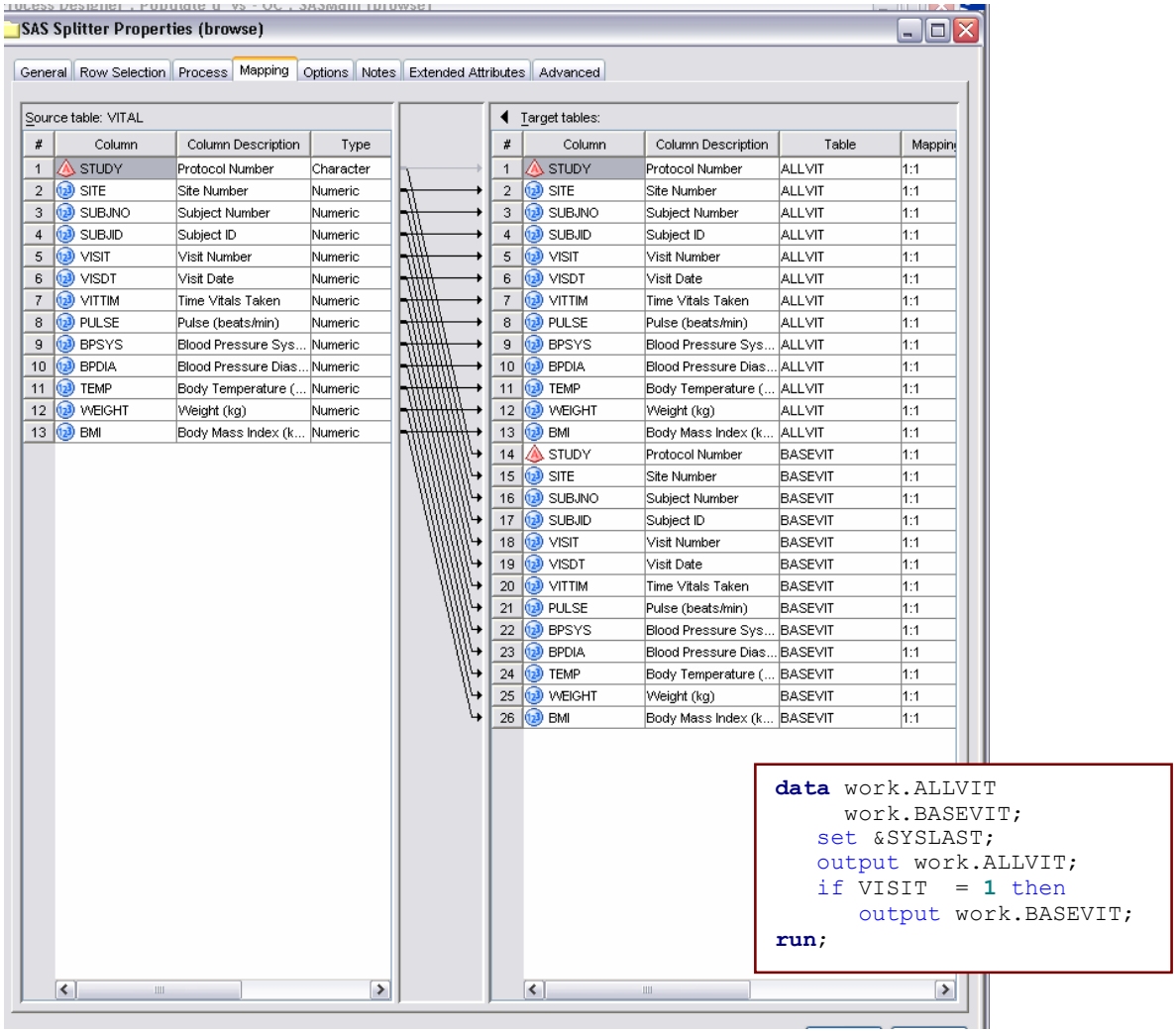


Figure 9 An example SAS Splitter process in DI Studio used for splitting a dataset into two or more datasets.
In this example, there was a requirement for a change from Baseline to be reported, and the easiest way to specify that was to split the initial dataset into two. The Splitter task allows the definition of subsets of data and then mapping as shown above to the two or more target datasets.

The two targets can then be joined together using a join as seen earlier to get the Baseline information on the same rows as visit specified data so the calculations can be performed.

Map. Once the data is joined and manipulated as desired, the mappings can be completed to achieve the target dataset. Mappings can take place in several places throughout the transformation process – when creating the change from baseline variables, they can be mapped in the splitter or SQL join. For example renaming baseline Pulse to B_PULSE can be done in these processes. For more complex mappings, these can be done in the final Load step.

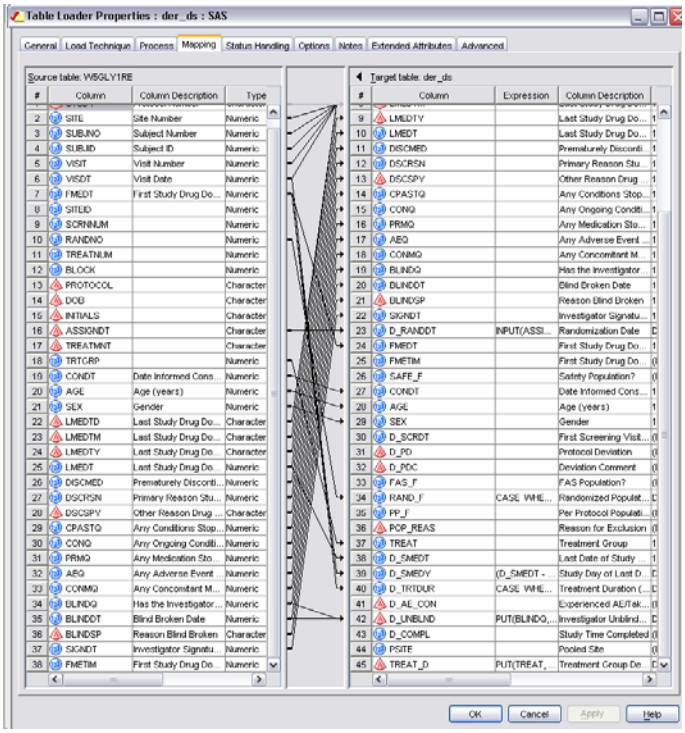


Figure 10 An example Mapping in DI Studio.

The mappings can be on a 1:1 basis or derived. When a variable has a direct mapping, it is easy to specify but if the target variable needs to be derived then an expression needs to be built. More than one original variable can be mapped to a derived variable and this is necessary if they are used to build an expression. In Di Studio it is easy to map variables, just simply drag and drop to the target. The generated code is below:

```
proc sql;
  create view work.mapped as
    select STUDY length = 20,
      SITE length = 8,
      SUBJNO length = 8,
      SUBJID length = 8,
      VISIT length = 8,
      (INPUT(ASSIGNDT, DATETIME23.)) as D_RANDDT length = 8 format = DATE9.,
      FMEDT length = 8,
      AGE length = 8,
      SEX length = 8,
      (CASE  WHEN RANDNO^= .
             THEN 1
             ELSE 0
       END) as RAND_F length = 8 format = YNNUF.,
      . as PP_F length = 8 format = YNNUF.,
      "" as POP_REAS length = 200,
      TRTGRP as TREAT length = 8 format = TREAT.,
      LMEDT as D_SMEDT length = 8,
      ((D_SMEDT - FMEDT)+1) as D_SMEDY length = 8,
      (CASE
         WHEN D_SMEDT=. THEN (VISDT-FMEDT)+14
         WHEN (D_SMEDT=. and VISDT = .) OR FMEDT=. THEN .
         ELSE (D_SMEDT-FMEDT) +1
       END) as D_TRTDUR length = 8,
      (PUT(BLINDQ, YNNUF.)  || PUT(BLINDDT,DATE9.)) as D_UNBLND length = 13,
      (PUT(TREAT, TREAT.)) as TREAT_D length = 15
   from &etls_lastTable;
quit;
```

Instead of using a straightforward if-then-do loop in base SAS code, DI Studio uses a case-when-then loop instead. There are several elements of DI Studio that are SQL based and it is necessary to be familiar with these aspects before embarking on DI Studio transformations.

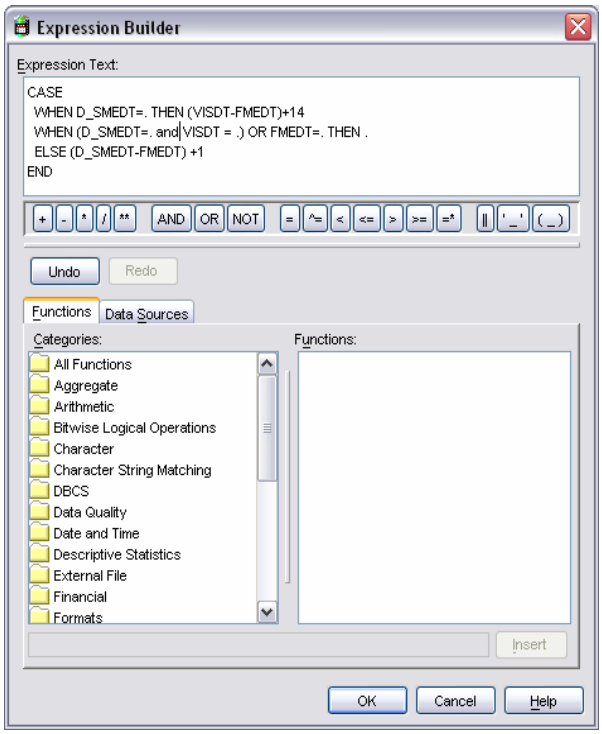


Figure 11 An example Case-When-Then expression built in DI Studio

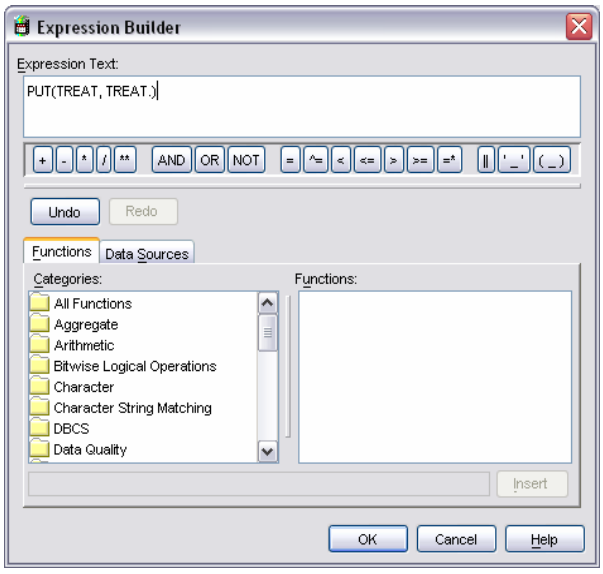


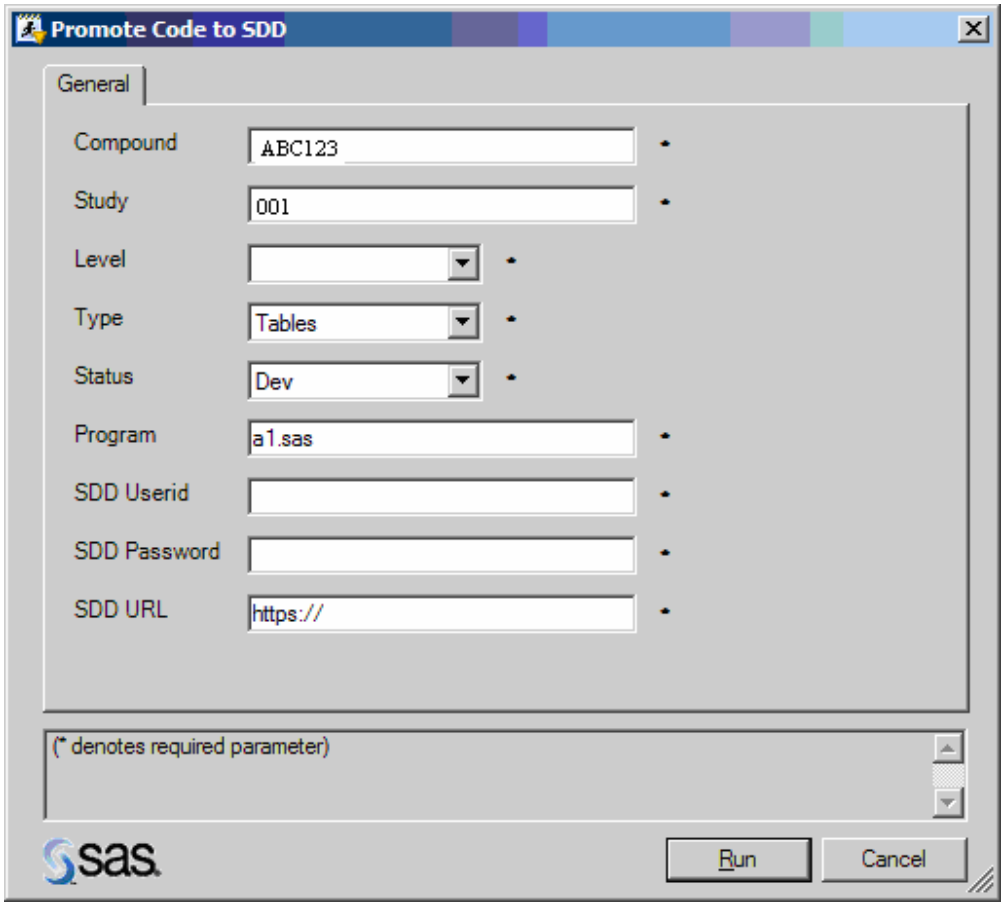
Figure 12 An example simpler mapping – putting numeric variables to character using a put( ) function and applying it's format.

For extremely complex transformations that cannot be done using a pre-defined transformation it is possible to enter a user-written transformation. Although written in Base SAS code, the final results are then loaded or mapped using the regular DI tools. So the metadata can still be traced through to the final dataset, although less easy to see the expressions and mappings. Once the variables have been mapped as appropriate, the testing phase starts. It is an iterative process to iron out issues but once submitted, the code is generated and submitted to the server interactively. The process can be saved and checked in to the study repository using a dev or qc area depending on which programmer was performing the process. In order to continue the cycle to SDD the source code should be opened and saved to the remote SAS server.

STEP 4 – LOCAL SERVER ENVIRONMENT

The replicated Raw data to supply metadata for the transformations and to populate the target datasets when the code is submitted is stored on the local Server Environment. When the code is exported from DI, it is held on the local Server environment.

A stored process to promote exported code to SDD is stored and accessed on the Local Server by using a BI tool such as Enterprise Guide. The stored process should convert any aspects necessary for the code to run in the SDD environment and supply a setup program in the SDD destination folder if it doesn't already exist.




```
data _null_;
   length _text_ $ 4096;
   infile dspathA LRECL=4096 length = len;
   input _text_ $varying4096. len;
   if index(lowcase(_text_), 'libname') then
   do;
      if (index(lowcase(_text_), 'sdb') or index(lowcase(_text_), 'ddb')) and not flag then
      do;
         _text_ = '%include &setup;';
         flag + 1;
      end;
      else delete;
   end;
   file dspathB LRECL=4096;
   put _text_;
 run;
```

STEP 5 – SDD ENVIRONMENT

The promoted code is stored in SAS Drug Development and enters a Quality Control phase.

Upon promotion of the code a setup.sas program can be created that sets up the libname statements in a way that SDD can utilise, this is automatically set in the program. The QC steps can be performed by executing both a primary and QC program in the SDD environment. A Compare procedure can be executed to validate the two different sets of output produced. If there are any discrepancies, the cycle should be repeated until both primary and QC programmers are satisfied that the programs are complete.

The primary program can be promoted to production in SDD and should have version control enabled so any future versions do not overwrite the current version, but merely increment the version. Previous versions are still accessible and if necessary can be reverted to through an export and save process. The code can also be electronically signed by any number of parties, but would recommend the programmer and QC person at a minimum, including a user (who may in fact be the QC person).

Once the QC step is complete, and versioning enabled, the code is executed to produce the final output – in this example, derived datasets. This produces logs and output and is available in a complete package that even wraps up the input and output data to enable successful recreation at a later date. The code can be executed using a menu feature from the main screen in SDD or from within the Process Editor. The final results are therefore in a validated state as all testing takes place in a local environment before promotion and are electronically signed and version controlled.

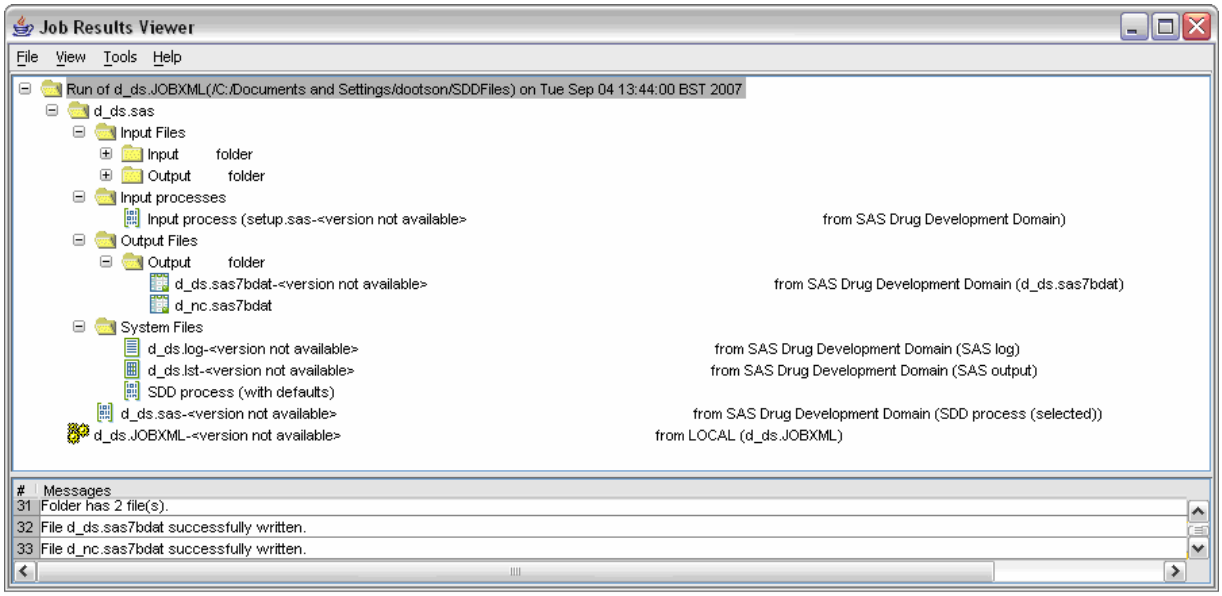


Figure 13 An example Job Results View from executing a program in SAS Drug Development

CONCLUSION

This paper does not profess to use all available functionality of the SAS tools available, but more use them effectively in the statistical environment to provide a validated framework. DI Studio is traditionally used to create data marts with dimensions, whereas here it is used to create suites of programs that will convert standard to derived data. The main benefits include documentation, validation and security. Notes and attachments can be added to any processes in DI Studio that can provide further information in the generated code, this might include specifications or descriptions of very complex transformations. The graphical flow can be saved as a gif file and inserted into any documentation. Most importantly the impact of changes to any metadata variables can be traced.

True, that using EG to create standard outputs or case-when-then statements to manipulate data may not be most favourable for an experienced programmer, but with practice comes a certain amount of familiarity and efficiency. Other benefits mentioned will outweigh an initial period of learning the new processes.

SAS Drug Development provides a framework to keep the final code and outputs, documents and standards validated using version control and electronic signatures. Further benefits include the ability to join data together quickly and easily to respond to regulatory requests if necessary. Study definitions in SDD can provide an easy look across many similar studies for the same reason.

The change to such a system and processes may come as a shock to the traditional types of programmers as the pure coding element somewhat disappears, although creating tables and deliverables still utilises tools such as base SAS and Enterprise Guide. One of the hardest aspects is change management as many traditional SAS users will not like the prospect of GUI interfaces and will want to remain coding in Foundation SAS or even in some cases a plain text editor! However, the benefits of a lifecycle are clearly demonstrated here. Security, Traceability, Documentation, Global working, Easy access to global data, efficiencies brought by standards, easy maintenance of other's code due to better documentation and global working, management of data and code, and finally a very defendable audit position.

ACKNOWLEDGMENTS

The author would like to thank Ron Coleman and Dave Smith of SAS for their input to the process and help along the way.

BIOGRAPHY

Alistair Dootson, SAS Consultant
Alistair Dootson is an independent SAS consultant who specializes in the more technical features of SAS, such as the implementation of tools and new technology. Alistair has spent more than 10 years in the pharmaceutical industry working in BDM groups and in IT as a Clinical Support Manager in a major CRO before working as an independent consultant.

CONTACT INFORMATION

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

Alistair Dootson
Dootsonic Ltd
Phone: 07824 364958
Email: ali_dootson@hotmail.com
Web: