Migrating 4GL SAS Code to SAS 9 EBI and Enterprise Guide

Michael Needham, Amadeus Software Limited

ABSTRACT

For many organisations, the decision to migrate from 4th Generation SAS code represents not only a considerable financial investment but also typically requires the migration of existing code. What are the challenges faced when migrating fourth generation SAS code to a SAS 9/EBI and Enterprise Guide environment? This paper explores the challenges and opportunities presented by this process; it describes a practical approach to the overall process; and provides an example of some of the tools developed by Amadeus to help facilitate the process.

INTRODUCTION

This paper outlines a general approach that might be adopted when faced with the task of migrating 4th generation code to the EBI and Enterprise Guide environment. At the outset, it is important to recognise that no wholly standardised approach can ever be rigidly defined since many aspects of each code-migration project are going to be different; the volume of code to be migrated, the nature of existing code, the type and quality of data sources and the nature of the organisation itself will all affect the best approach to migration task being undertaken.

One of the strengths of Enterprise Guide is the Project Designer – allowing the user to see a flow of tasks within each workflow. It is generally possible to rebuild 4th generation code as an EG project but there is no simple way to automate this process. Any automated code migration process must therefore focus on updating the code to enable it run successfully in the new environment, a major change such as migrating code to SAS EBI is bound to present a cultural challenge and can be expected to take some time; successfully encouraging the users to adapt to the new environment typically then depends on effective training and encouragement.

This paper does not provide a guide to setting-up and configuring SAS 9 EBI or to Enterprise Guide, although some references to these have been included at the end of this paper. It has also been assumed that data migration has been completed before code migration begins.

PLANNING/CONSULTATION PHASE

The first task that must be undertaken in any project of this type must be a planning/consultation phase, this is likely to be the first opportunity to see examples of the code to be migrated and therefore to set user expectations. Migrating to the EBI environment is likely to present the organisation with an unsurpassed opportunity to standardise and control code as well as data, so the planning phase is the time agree how far the organisation wants to undertake this progress; the goal may be anything from simply migrating existing code and testing in the new environment through to developing macros, stored processes, building format libraries and cascading style sheets.
The situation can be expected to vary greatly between organisations; typically existing SAS code will be found to be a mixture of code, often saved on local machines and there may be a good deal of redundant code too, so getting a clear picture of the current state of code must be one of the primary goals at this stage.

It is easy to under-estimate the time required to complete the consultation process – it is unlikely that it will be possible to identify all of the code to be migrated at the first attempt and this means that identifying code for migration will probably be an iterative process. It is reasonable to expect two or three iterations before all the code for migration is finally settled, so agreeing on a deadline for each iteration at this stage is important and will help to set realistic timelines for the project.

It is also helpful to establish a primary point of contact in the organisation as well as key individuals with a good understanding of how things are currently managed; understanding who these individuals are is invaluable in bringing the project to a successful conclusion. In more regulated environments, the migration process will be controlled by Standard Operating Procedures (SOPs), so including someone from within the organisation with a QA background in the consultation process is essential.
Finally, consideration should be given to how progress will be monitored and reported to the organisation, in many cases a weekly summary to all interested parties will be sufficient but this aspect should certainly be agreed at this stage.

In common with any project of this type, decisions taken at this stage are going to affect every stage of the project, so getting general agreement on who is expected to do what and when it will happen will profoundly influence the chances of a successful outcome.

QUESTIONS TO BE CONSIDERED IN THE PLANNING PHASE

Are there multiple versions of code?

Is there a pre-existing structure in which code is saved?

Does the project include building an archive for redundant code?

Are hard-coded paths with drive letters used or full UNC paths?

Are user defined formats and informats defined?

Does code contain references to external file locations?

What controls are in place within the organisation?

How will the project progress be managed?

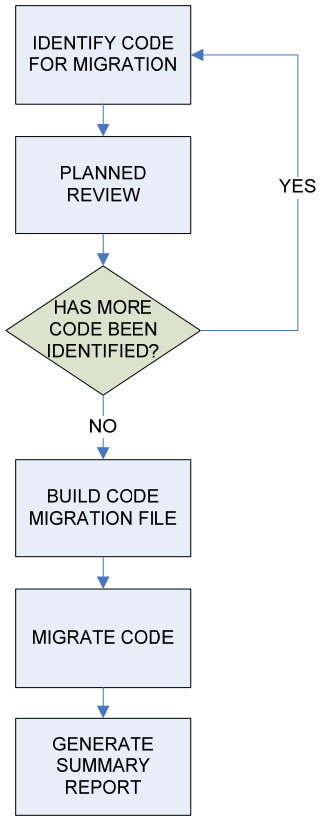
How much of the migrated code should be tested?

What should the success criteria be for testing

What training will users require to use migrated code?

Has the process of data-migration been completed?

The diagram below illustrates the structure of a code migration project:



THE CODE MIGRATION PHASE

Once agreement has been reached on how the project will progress and it has been possible to review examples of the code to be migrated, it may be possible to develop some tools to help you with the process of code migration. In this example we have used a SAS macro to help us automate the process. We will only provide an overview here of some of the tools developed by Amadeus to help with

this process, but sample code has been included in the Appendix of this document that could be adapted as required.

Before code can be migrated, a process of rationalising the data should have been undertaken so that data will ideally either be registered in meta-data or will be pre-assigned in an autoexec file. Changes to the migrated code will be needed to reflect this.

GENERAL CONSIDERATIONS BEFORE CODE MIGRATION:

The following points should be considered before migrated code is implemented as a code node with Enterprise guide:

1. Hard-coded paths and drive letters must be replaced by full UNC paths;
2. ODS statements may contain hard-coded paths;
3. Reserved statements and functions must be replaced or modified as required;
4. Use of user-defined formats and macros.

We will consider each of these aspects in more detail:

THE CODE MIGRATION PHASE IN DETAIL.

(1) Hard-coded paths are typically found in these statements and functions:

LIBNAME statements;

FILENAME statements;

FILE statements;

INFILE statements;

%INCLUDE statements;

PATHNAME function;

MACRO variables.

Paths should be replaced with full UNC paths, so that '\\fileServer01\Lev1\department\project\data\sub_folder' should be used, rather than using a drive letter. In the SAS EBI environment data will have been migrated and registered in metadata so there should be no need to register libraries directly in code.

(2) ODS statements – where these direct output to a specific location, the path should be re-coded to use the full UNC path, in Enterprise Guide, it may be better to remove the ODS statement altogether and use the default Enterprise Guide Output.

(3) Reserved Statements and functions – By default, SAS Enterprise Business Intelligence server has the setting for system calls as NOXCMD – this disables a number of SAS functions, a possible strategy to adopt in each case is presented in table 1 below:

(4) User defined formats and macros should run without changes, although the addition of FMTSEARCH and SASAUTOS options to the autoexec file may also be needed.

(5) Full screen functionality – since SAS EBI sessions function as services, the statements in the following table cannot be run in SAS EBI/Enterprise Guide, the table below presents a possible strategy that might be adopted in each case:

Table1 – Possible Coding Stategies

SAS Statement or Function	Circumvention
PIPE and NAMEPIPE device types on the FILENAME statement	Review function performed at operating system level for recoding e.g. as data step
CALL SYSTEM routine	Review function performed at operating system level for recoding e.g. as data step
X command	Review function performed at operating system level for recoding e.g. as

	data step
Dynamic Data Exchange (DDE);	Convert to output delivery system TAGSETS.EXCELXP.
%SYSEXEC macro	Review function performed at operating system level for recoding e.g. as data step
SYSTASK statement	Review function performed at operating system level for recoding e.g. as data step
FILENAME function	Consider filename statements
Proc FSEDIT	Where data entry is required the standard data editor within Enterprise Guide can be used.
Viewtable	Where data entry is required the standard data editor within Enterprise Guide can be used.
%WINDOW, %DISPLAY	Consider creating a stored process.

USING A MACRO TO MODIFY CODE FOR MIGRATION.

In this paper we will present an example of a scenario in which there is a large volume of code to be migrated so that it will be efficient to utilise macros to speed up migration.

In this example, code migration is broken down into three main steps, each of which uses its own macro:

1. The first macro identifies all files from a chosen source folder with a '.SAS' suffix; then parses each file in turn to look for statements containing words in each line that indicate that modification may be required; then builds a .CSV file that includes these lines, columns for text to be replaced and a column for replacement text. The .CSV file built by this macro can be edited and provides the input to the second macro.
2. Run a find/replace macro to implement the edits made in the .CSV file.
3. Run a macro compare programs before and after modification and generate a report to confirm the process has been completed successfully.

Below is a brief description the functioning of each macro:

STEP 1 - SCAN THE CODE FOR KEY WORDS

This macro includes parameters for the location to search for SAS files, the location to save the .CSV data to and a list of keywords to search for (using a separator for keywords). It will perform the following tasks:

1. Scan the source directory.
2. Identify any filenames with a .SAS suffix.
3. Read in each line of the identified .SAS files and scan for keywords.
4. Retain any lines of code in which the key-words are found and use these to build an output dataset.
5. Write the output dataset to a .CSV file that can be edited for use by the next macro.

Below is the macro call and part of the csv output generated by this macro (the complete macro has been included as Appendix A):

```
%program_search (scan_dir=%nrstr(C:\Migration\Source Programs),
                 output_dir=%nrstr(C:\Migration\Output),
                 find=%nrquote(LIBNAME*FILE*%INCLUDE*PATHNAME*BODY=*PIPE*CALL SYSTEM*DDE));
                );
```

All Programs.csv

	A	B	C	D	E	F	G	H	I
1	Folder	Program	Full Path	Migrate Y/N	Statement	Update Y/N	Change From	Change To	New Folder
2	C:\Migration\Source Programs	Data_Read.sas	C:\Migration\Source Programs\Data_Read.sas	Y	libname amadeus "C:\Data";	Y			
3	C:\Migration\Source Programs	Data_Read.sas	C:\Migration\Source Programs\Data_Read.sas	Y	filename sales1 "C:\Data\Raw\File1.txt";	Y			
4	C:\Migration\Source Programs	Data_Read.sas	C:\Migration\Source Programs\Data_Read.sas	Y	infile sales1;	Y			
5	C:\Migration\Source Programs	Data_Read.sas	C:\Migration\Source Programs\Data_Read.sas	Y	infile "C:\Data\Raw\File2.txt";	Y			
6	C:\Migration\Source Programs	Reports.sas	C:\Migration\Source Programs\Reports.sas	Y	libname amadeus "C:\Data";	Y			
7	C:\Migration\Source Programs	Reports.sas	C:\Migration\Source Programs\Reports.sas	Y	ods csv file = "C:\Reports\Sales.csv";	Y			
8									

All Programs

Keywords found in Data_Read.sas

Statements Containg Keywords
libname amadeus "C:\Data";
filename sales1 "C:\Data\Raw\File1.txt";
infile sales1;
infile "C:\Data\Raw\File2.txt";

Keywords found in Reports.sas

Statements Containg Keywords
libname amadeus "C:\Data";
ods csv file = "C:\Reports\Sales.csv";

STEP 2 – FIND/REPLACE

Once code containing the above keywords has been identified and reviewed as necessary, the CSV file generated by the first Macro can be modified and then submitted as input to a second macro to replace text and to copy and write code to a new output location.

Data_Read

```
libname amadeus "C:\Data";
filename sales1 "C:\Data\Raw\File1.txt";

data amadeus.sales1;
   infile sales1;
   input product $ category $ value;
run;

data amadeus.sales2;
   infile "C:\Data\Raw\File2.txt";
   input product $ category $ value;
run;
```

Data_Read

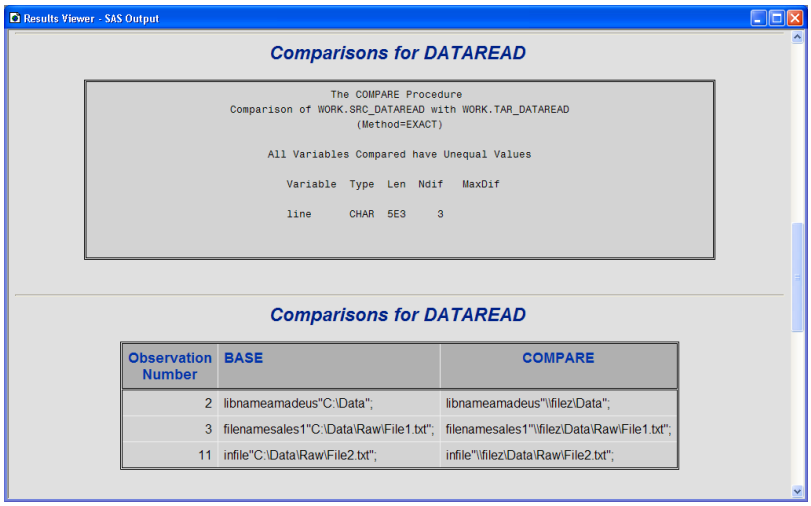
```
libname amadeus "\\filez\Data";
filename sales1 "\\filez\Data\Raw\File1.txt";

data amadeus.sales1;
   infile sales1;
   input product $ category $ value;
run;

data amadeus.sales2;
   infile "\\filez\Data\Raw\File2.txt";
   input product $ category $ value;
run;
```

STEP 3 – REPORT CHANGES

Finally, the original and modified code are compared and a report generated that summarised the differences between each version of the code. This provides evidence that the all the code has been migrated:



OTHER CONSIDERATIONS

There are a number of other considerations when migrating code to the EBI or Enterprise Guide Environment. Migrated code:

In Enterprise Guide the default option for the VALIDVARNAME option is 'ANY' (allowing the use of spaces and all characters in SAS names), in base SAS the default for this option is 'V7' (where this is not allowed). In general, this means that code written in 4th Generation SAS should run without modification in the new environment; it is worth considering setting this system option to 'V7' however to maintain this standard.

POST-MIGRATION TESTING

Once the migration process is complete, a sample of code will need to be tested; the type and method of testing should have been agreed in the project planning phase. The proportion of programs to be tested will typically depend on the function and number of programs but it clearly makes sense to test all programs with critical functions and those which are used most regularly. Identifying these programs is best carried out by the user and where possible, the program owners should undertake this process themselves.

Where circumstances allow, an alternative approach would be to run existing code in parallel with migrated code, perhaps for a pre-agreed duration or at to run each example of migrated code at least once. This approach can be run in addition to testing a random selection of migrated code.

It is not unusual to find code that cannot be re-run (for example, where code builds a monthly table that cannot be over-written) and, in this case, temporarily re-naming the output dataset may be necessary.

CRITERIA FOR SUCCESS

Criteria for success is also one of the aspects that should have been agreed in the project planning phase and will depend, to a large extent on the nature of the organisation and purpose of the code; a number of approaches can be adopted:

An ERROR and WARNING free log is something that is clearly desirable but it is potentially unwise to use this as a test, unless the original code could also satisfy this test (this may often not be the case), especially in a less regulated environment where many individuals each have their own code.

In other cases allowing no new errors or warnings but allowing pre-existing warnings to remain may be acceptable.

In more highly controlled environments a direct comparison of outputs may be appropriate, using PROC COMPARE as well as searching for ERRORS and WARNINGS.

UNDERTAKING THE MIGRATION PROCESS IN HIGHLY REGULATED ENVIRONMENTS

The level of Validation/Verification of SAS code required can be expected to vary greatly from one organisation to another and will probably also depend on what the purpose of the code. In many cases, if the purpose of the program is essentially to explore data, then it

may often be sufficient that the code runs cleanly, without warnings or errors in the log. In highly regulated organisations, such as in the pharmaceutical industry and where critical decisions will be made on the results of code, FDA and/or other regulations are likely to apply and verification of code will be a minimum requirement for code that is used only once through to full validation for code that will be run many times will almost certainly be necessary.

The interpretation of such regulations is likely to vary greatly between organisations but must be considered in planning the migration work; time spent in understanding how regulations are interpreted within an organisation will, at the very least, pay dividends in running the process efficiently, but more importantly will help to ensure that the process is conducted in an acceptable way such that work will not have to be repeated; the worst possible outcome. In highly regulated organisations, it is helpful to understand that organisations like the FDA take a risk-based approach, so that evidence of a risk assessment will be expected and that all required steps identified by the assessment have been taken.

If consideration of regulatory requirements is carried out at an early stage in the project, the chances of a trouble-free code migration will be greatly improved.

CONCLUSION

The process of code migration typically represents only one step in the larger process of moving to SAS9 EBI and Enterprise Guide and the approach taken here represents only one possible route. The specific example presented here represents an approach that could be easily adapted as required and has been found to be cost-effective and able to generate usable code.

In this paper, I have attempted to demonstrate that the task of code migration presents both a challenge and an opportunity in organisations. The challenge is to identify, document and then effectively implement a strategy that is appropriate to the organisation. Undertaking this process presents an organisation with the opportunity to standardise and improve the quality and consistency of SAS code within the organisation. Successfully completing this process can not only improve the quality of SAS code by making it more robust, but can also be an important step towards making code compliant.

REFERENCES

SAS Support general guidance on migration to the EBI environment: http://support.sas.com/rnd/migration/

SAS Support guidance on code migration: http://support.sas.com/rnd/migration/planning/software/programs.html

Considerations for Pharmaceutical Code Migration: http://support.sas.com/rnd/migration/industry/industry_pharma.html

Links to FDA guidance http://www.fda.gov

CONTACT INFORMATION

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

Author Name	Michael Needham
Company:	Amadeus Software Limited
Address:	Mulberry House, 9 Church Green, Witney, Oxon OX28 4AZ
Work Phone:	+44 (0) 1993 848010
Email:	Mike.Needham@amadeus.co.uk
Web:	www.amadeus.co.uk

APPENDIX A – MACRO TO IDENTIFY CODE FOR REPLACEMENT

```
%macro program_search (scan_dir= ,find=, output_dir=);

  options mprint;

  %* Scan a directory and save all file names to macro variables;
  %let filrf=mydir;
  %let rc=%sysfunc(filename(filrf,%superq(scan_dir)));
  %let did=%sysfunc(dopen(%superq(filrf)));
  %let num_files=%sysfunc(dnum(&did));
  %if &num_files > 0 %then
  %do;
    %do i = 1 %to &num_files;
      %global file&i;
      %let file&i = ;
      %let file&i = %nrquote(%qsysfunc(dread(&did,&i)));
      %put &&file&i;
    %end;
  %end;
  %let rc = %sysfunc(dclose(&did));

  %* Create a dataset containing only .sas program names;
  data files;
    length program $100;
    %do j = 1 %to &num_files;
      program = "%nrquote(&&file&j)";
      if upcase(scan(program,-1,".")) = "SAS" then
      do;
        count + 1;
        call symputx("sasfile"!!compress(count),program);
        output;
      end;
    %end;
  run;

  %do m = 1 %to &num_files;
    %let sasfile&m = %superq(sasfile&m);
  %end;

  %* Count how many .sas programs there are;
  %let dsid = %sysfunc(open(files));
  %if &dsid ne 0 %then %let num_sasfiles = %sysfunc(attrn(&dsid,NOBS));
  %let dsclose = %sysfunc(close(&dsid));

  %* Calculate how many keywords are being searched for;
  %let num_search = %sysevalf(%qsysfunc(countc(&find,%str(*))) + 1);

  %* Save the keywords to macro variables;
  %do k = 1 %to &num_search;
    %let find&k = %qscan(&find,&k,*);
  %end;

  %* Open each program in turn and scan for the keywords;
  %do l = 1 %to &num_sasfiles;
    data program&l;
      length folder $400 program $100 full_path $500 migrate $1 line $5000 update $1 chg_from $500
         chg_to $500 new_folder $400;
      infile "&scan_dir.\&&sasfile&l" lrecl=5000 missover pad;
      input line $1-5000;
      program = "&&sasfile&l";
      folder = "&scan_dir";
      full_path = "&scan_dir.\&&sasfile&l";
      migrate = "Y";
      update = "Y";
      if not missing(line);
      if %do k = 1 %to %sysevalf(&num_search - 1);
           index(upcase(line),"&&find&k") > 0 or
```

```
        %end;
      index(upcase(line),"&&find&num_search");
    run;

    %* Save information for all sas programs into one dataset;
    proc append base = all data = program&l;
    run;

    %* Output HTML report for each program analysed;
    ods listing close;
    ods html file = "%nrquote(&output_dir)\Report - %qscan(&&sasfile&l,1,.).html";

    title "Keywords found in %nrquote(&&sasfile&l)";
    proc report data = program&l nowindows;
      column line;
      define line / "Statements Containg Keywords";
    run;

    ods html close;
    ods listing;
  %end;

  %* Create a unique list of programs to be updated;
  proc sql;
    create table migrate_update as
    select distinct program
    from all;
  quit;

  %* Identify programs in the scan_dir that are to be migrated but not updated;
  proc sql;
    create table migrate_no_update as
    select a.program,
    case
      when a.program = b.program
      then "Y"
      else "N"
    end as update
    from files as a left join migrate_update as b
      on a.program=b.program
        where calculated update = "N";
  quit;

  %* Create a list of all programs to be migrated;
  data all1;
    set all (in=a) migrate_no_update (in=b);
    if b then
    do;
      folder = "%nrquote(&scan_dir)";
      full_path = "%nrquote(&scan_dir)\"!!program;
      migrate="Y";
    end;
  run;
  %* Output csv file containing all statements in all programs needing editing.
     This file will be in the input for the find_replace macro;
  ods listing close;
  ods csv file = "%nrquote(&output_dir)\All Programs.csv";

  proc print noobs data = all1 label;
    label program = "Program"
          line = "Statement"
          folder = "Folder"
          full_path = "Full Path"
          migrate = "Migrate Y/N"
          update = "Update Y/N"
          chg_from = "Change From"
          chg_to = "Change To"
          new_folder = "New Folder";
  run;
```

```
  ods csv close;
  ods listing;

%mend;
```