Paper DM09-2012

A Basic Recipe for Building a Campaign Management System from Scratch: How Base SAS®, SQL Server and Access can Blend Together

Tera Olson, Aimia Proprietary Loyalty U.S. Inc., Minneapolis, MN

ABSTRACT

Designing and building a custom campaign management system can be a daunting task. The system we envisioned had to manage billions of rows of data, execute all of the components of a strategic marketing program, and provide a user-friendly interface for project management. During the build we learned that success depended on clear requirements for the design team, a thoughtful and organized system architecture, and efficient, tested and documented SAS code tying it all together. This paper provides information, both practical and technical, around building a campaign management system from scratch using Base SAS, Microsoft SQL Server and Microsoft Access. It addresses what worked well and lessons learned in hopes that others who are considering building such a system can benefit from the experience.

THE ISSUE AT HAND

Campaign management systems running on outdated mainframes are cumbersome and expensive to operate and maintain. They are especially difficult environments for data management and application development. When high-level project management data is stored in non-user-friendly applications, data entry can be duplicative and disjointed across varying departments of an organization. Creating a custom-built system would allow us to manage billions of rows of data, successfully execute all aspects of marketing campaign management, and provide a user-friendly interface for project management. Our company already had the components, expertise and tools in our analytic department to create a custom-built solution to meet our specific needs and demands. This saved time and resources that could have been spent on off- the-shelf options that would need to be customized later.

THE RECIPE FOR BUILDING THE SYSTEM

It is crucial to have a team who works well together and understands requirements fully. Time spent upfront on requirements gathering, architecture, and design is necessary. SAS code should be efficient and eloquent. Testing is crucial and automation is complex. Following are the details on how we went about building our campaign management system from scratch.

THE TEAM

The team building the system consisted of client services staff who understood the business requirements and were tasked with managing the project, an IT resource who knew the current system and what needed to be converted, and two SAS programmers with a combined 30 years of experience. One SAS programmer was the architect and main programmer while the other translated business needs, tested the system, and programmed as needed. A data analyst with MS Access experience built Access screens and VB functionality as well as set up the SQL server environment. To round out the team, other support people like higher level budget owners and a resource for hardware/software help were included.

It is important to define owners and agree on roles and responsibilities during the build. It is recommended that responsibilities be reviewed at the beginning of the build as the timeline is set and periodically during the build process. The roles of those who will manage and run the system once it is finished should be agreed upon before the system is complete. This will probably include employees who did not participate in the system build process, but have skill sets to manage and use it. Post-build roles include functions like Platform Manager, Campaign Specialist and Implementer, and Database Administrator.

REQUIREMENTS

- Assess existing situation
  During our build, we had the advantage of referring to a legacy mainframe campaign management system which had been in place for a number of years. Although it was disjointed and expensive, the framework provided an opportunity to change the components that didn't work and keep the ones that worked well. In our case, some of

the data structure from the old environment was kept but the technology and code was built from scratch. Inputs/outputs were already set and just needed to be modified to fit the new system.

- Assess what was brand new
  All code, process and most database tables would be built from scratch. The technology was within our department already, but hadn't been blended together in this way before.
- Collect what the end users wanted
  Down to each small detail, it's important to gather and discuss and gather and discuss some more with those who knew requirements.
- Create a timeline, revisit it frequently, and adjust as needed

DESIGN AND BUILD

The tool's design should be thoroughly planned and discussed before the build starts.

- Inputs and Outputs
  In our build, standard input and output file layouts were created and followed.
- Data
  At the most basic level, all data elements were listed and organized including necessary tables and variables. Hierarchies of the tables and what variables fit in each table were agreed upon. We thoroughly reviewed the tables to make sure there were not duplicate pieces of information and tables were as basic as possible without losing important information.
- Code
  System functionality was evaluated as well as what the coded needed to accomplish. Each SAS program was designed to do one major campaign step (loading inputs, and selection and creating mail files are a few examples).
- Process
  Process related questions were answered such as: When moving through a campaign lifecycle, what are the steps? What SAS programs will run when? How will SAS work with SQL Server efficiently?
- Screens
  Data entry screen functionality was assessed and designed much like the SAS code was evaluated and configured. Data entry needed to be simple for the end user, yet capture and organize the vast amount of information to run a campaign.

ADDITIONAL BUILD DETAILS

Along with constructing the database tables and writing the code and building the process and screens, a few other steps in the build were considered:

- Historical Data Conversion
  Master customer and campaign level tables were converted from the old system (Mainframe and Access) to the new tables. A map was created to show how the old fields and tables translated to the new ones.
- Testing
  Old campaigns that could be used for testing were identified. Inputs and expected outputs for the test campaigns were already known. Code was built and each step in the campaign process was tested to make sure outcomes were the same between the old and new system. The old system was not "turned off" until new outputs matched the old ones. The most important components to be tested would be:
    - SAS code
    - Screens
    - The process, or flow of the system
- Documentation
  Time and budget should be allocated for documenting the various aspects of the system:
    - System schematics
    - Data dictionary
    - Database business rules
    - Process Flow with Controls
    - Detailed SAS code documentation
    - Start to end campaign instructions, "How to Run a Campaign"
    - Revamped documentation on campaign reporting and results library

THE CAMPAIGN MANAGEMENT SYSTEM

The campaign management system was built in about 11 months. It exists in a Windows 2008 64-bit environment, currently using SAS 9.2. All code is in Base SAS, utilizing a good deal of PROC SQL, SAS Macro programming, and ODS for audit reports. The new system has approximately 10 SAS programs defined by each major campaign step.

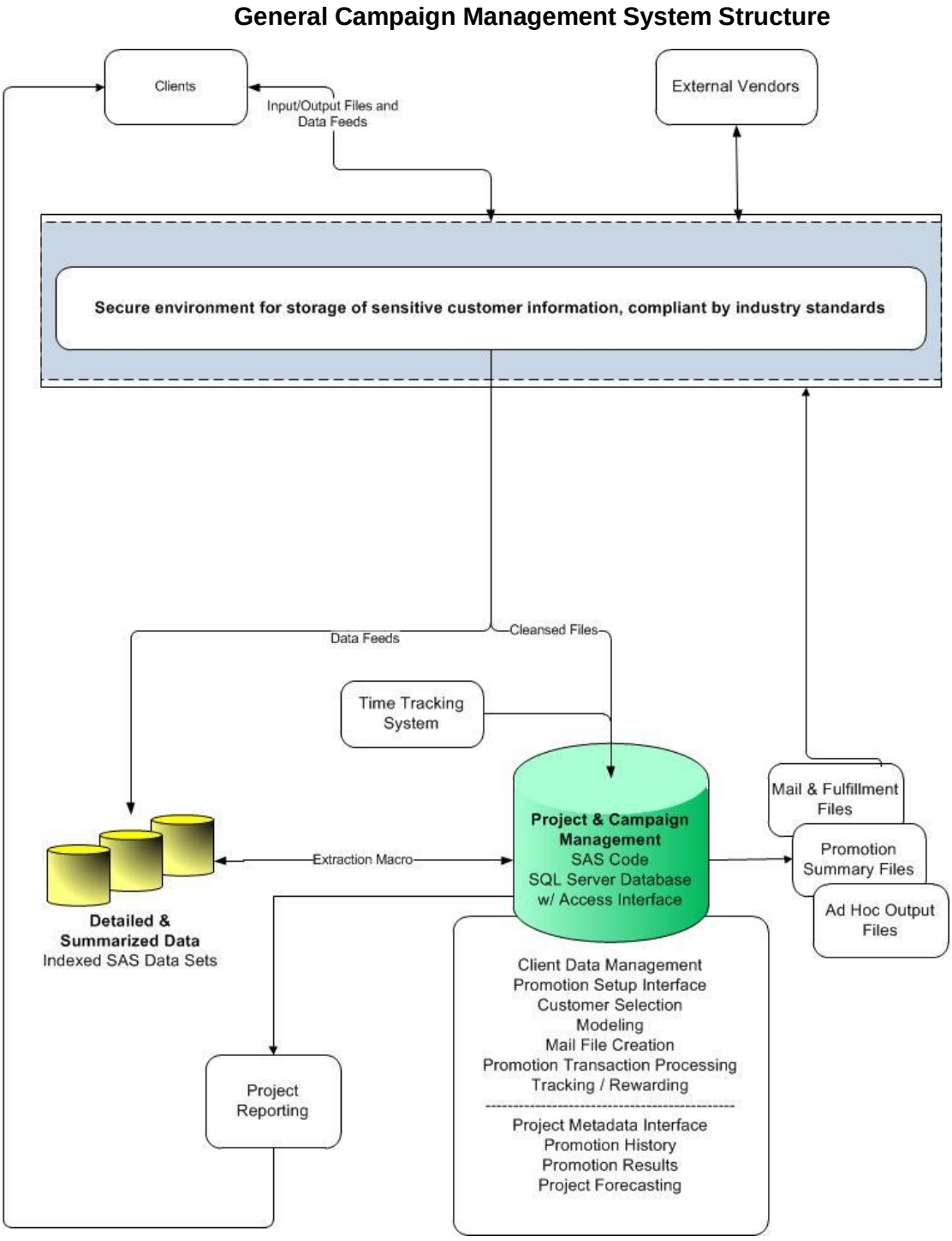


Figure 1. Campaign Management System Structure

All campaign requirements are entered into Microsoft SQL Server tables through Microsoft Access data entry screens. There are about 50 screens, partitioned by campaign and project management function. A heavy amount of

Visual Basic runs behind the screens. Campaign tables are stored in SQL Server. They are not relational as no advantage of a relational structure was identified. There 40 tables and 60 reference tables. The campaign management system was built for specific business needs, but is very flexible and can easily have functionality added or removed. It flexes as the business needs change.

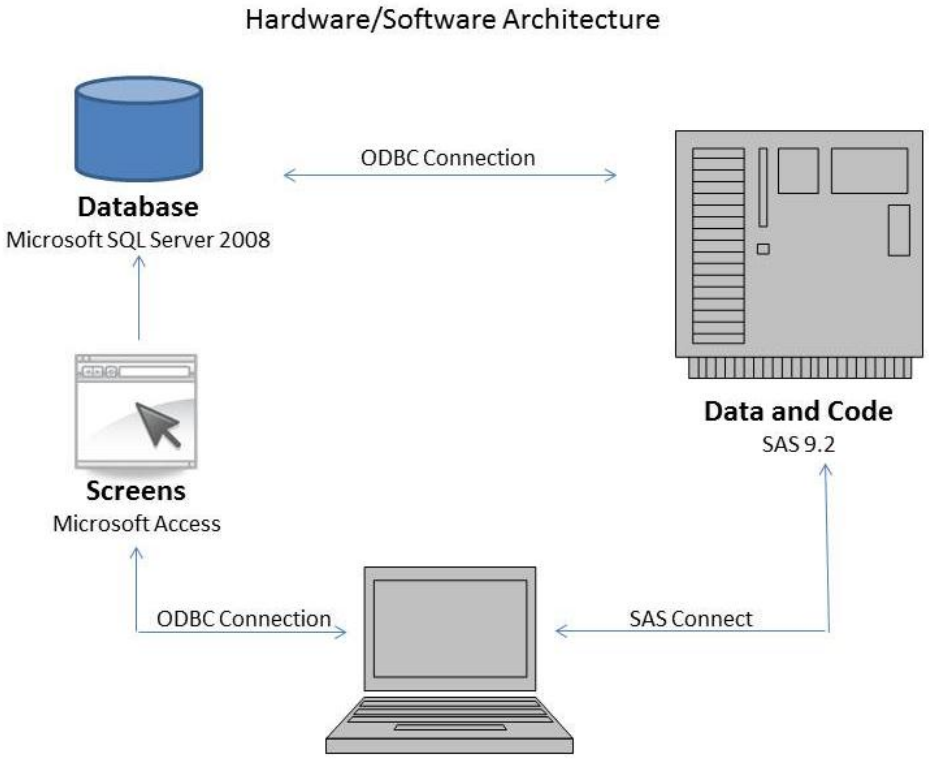


Figure 2. Hardware/Software Architecture

THE USAGE PROCESS

The high level steps behind the processes in our campaign management tool are similar to the following for each major campaign step:

- User enters metadata through the screens into the SQL Server tables including the Ref Job table
- A SAS program runs and pulls information and customer data from the tables including the Ref Job table
- SAS processes (often using the extraction macro) and manipulates
- SAS appends data back into the tables
- User moves onto the next campaign step and repeats the above

SAS AND SQL

We use SAS to process all data, reading from and writing to the SQL tables. SAS communicates with SQL server through an ODBC connection and a libname.

Loading to SQL Tables – A Basic PROC APPEND:

```
proc append base=vma_db.ds_calculated_fields data=NewFields&i;
run;
```

Extracting from SQL Tables – A Basic PROC SQL Step:

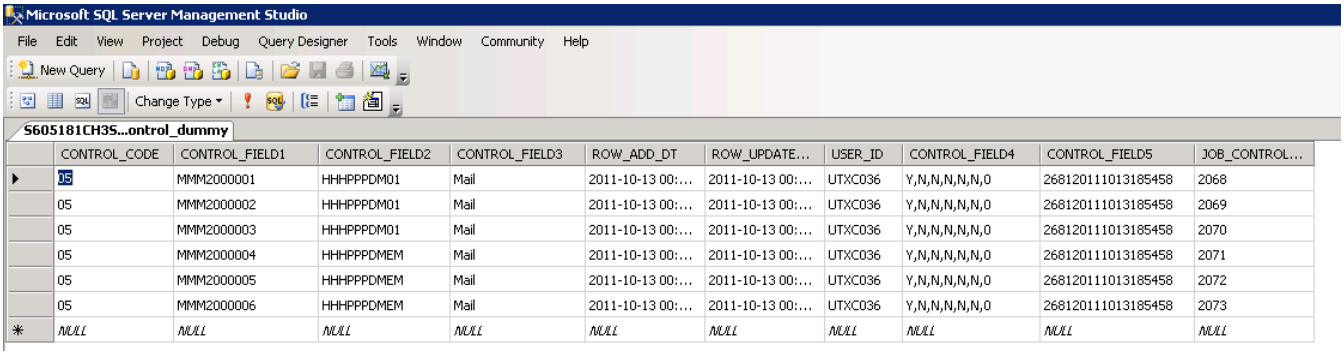
```
proc sql;
connect to odbc (dsn=&db user=&user);
create table ds_calculated_fields1 as
select *
```

```
from connection to odbc
(select * from ds_calculated_fields where p_id = &qP_ID);
disconnect from odbc;
quit;
```

REFERENCE JOB ('REF JOB') TABLE

A special 'Ref Job' table holds numbered commands to enable SAS to talk to the SQL database. There are several campaign functions that are captured in the Ref Job table and each function has a number. When the end user enters criteria for a function through the Access screens, the Ref Job table is updated. The Ref Job table contains crucial data points for each function that are read by SAS and translated. Each data point becomes a macro variable somewhere in the SAS code. Examples would be information for mail file headers or whether name and address should be included or excluded from the file's detail records.

```
data ref_job_control;
set vma_db.ref_job_control;
where CONTROL_CODE ='05' and compress(control_field1) in (&P_IDs);
Name = scan(control_field4,7,',');
Address = scan(control_field4,9,',');
if Name in ('Y','1') then Name = '1';
else Name = '0';
if Address in ('Y','1') then Address = 1;
else Address = 0;
run;
```



	CONTROL_CODE	CONTROL_FIELD1	CONTROL_FIELD2	CONTROL_FIELD3	ROW_ADD_DT	ROW_UPDATE...	USER_ID	CONTROL_FIELD4	CONTROL_FIELD5	JOB_CONTROL...
▶	05	MMM2000001	HHHPPPDM01	Mail	2011-10-13 00:...	2011-10-13 00:...	UTXC036	Y,N,N,N,N,N,0	268120111013185458	2068
	05	MMM2000002	HHHPPPDM01	Mail	2011-10-13 00:...	2011-10-13 00:...	UTXC036	Y,N,N,N,N,N,0	268120111013185458	2069
	05	MMM2000003	HHHPPPDM01	Mail	2011-10-13 00:...	2011-10-13 00:...	UTXC036	Y,N,N,N,N,N,0	268120111013185458	2070
	05	MMM2000004	HHHPPPDMEM	Mail	2011-10-13 00:...	2011-10-13 00:...	UTXC036	Y,N,N,N,N,N,0	268120111013185458	2071
	05	MMM2000005	HHHPPPDMEM	Mail	2011-10-13 00:...	2011-10-13 00:...	UTXC036	Y,N,N,N,N,N,0	268120111013185458	2072
	05	MMM2000006	HHHPPPDMEM	Mail	2011-10-13 00:...	2011-10-13 00:...	UTXC036	Y,N,N,N,N,N,0	268120111013185458	2073
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 3. Ref Job Table example

EXTRACTION MACRO

The system is very efficient and is able to track an unlimited number of campaigns at once. The only limitation is the personnel needed to enter data and run process steps. It can run an unlimited amount of customers through the system simultaneously. A full campaign can be run from start to finish in one work day if all requirements are entered.

We load over 100MM detail records each day and keep the data indefinitely. In order to store this volume of data in the finite amount of space available, all possible fields on the dataset are compressed into packed numeric fields. Because of this, the data must be extracted from the file in a packed format, and unpacked prior to use. Each day is its' own dataset and is indexed. (These files once used Generation Data Group (GDG) logic but this was problematic when files arrived or were loaded out of sequence, but that is possibly another paper.)

The system can pull 25MM detail records in seconds due to a SAS extraction program that was was built internally for the system. The extraction macro is read only and contains 28 macro variables that are entered and then called by the end user. The macro variables contain fields like input dataset name, fields to keep in the output dataset, date parameters and client specific parameters. About 8 team members use this macro on a daily basis.

Pack Example:

```
pk_zip=put(input(put(max(input(zip,5.),0),z5.),5.),pk3.);
```

Unpack Example:

```
zip=put(input(pk_zip,pk3.),z5.);
```

AUTOMATION

The automation of the full campaign management system, meaning schedule all jobs based on data entry and datetime triggers, was a logical next step. Our plan was to automate fully in Phase II of the campaign management system build. But once the time came to automate, the situation was assessed and it was decided that based on workload and staffing, we would only automate a portion of the system. The daily file loads and monthly source data creation were automated but not the actual campaign functions. This resulted in a savings of about 25 hours a month.

The concept behind daily file automation is fairly simple, but beautiful as it saves so much time. There are 3 main steps: 1) SAS program, 2) .cmd file and 3) a job in the Task Scheduler in Windows.

1. The SAS program is whatever code you want to run by itself and should be able to:
    - Run properly and successfully on demand
    - Comfortably use macro variables
    - Know what file sequence to read in (by saving sequence numbers in headers of datasets and then reading in those headers and incrementing every day)
    - Make sure the input file exists (using SAS functions)
    - Zip and delete the input file when done (using X commands in SAS)
    - Send out an email when done processing
2. The .cmd file contains syntax telling what SAS code to run and where to save the log and lst
3. The job in the Windows Task Scheduler calls the .cmd file which talks to the SAS program

The key in setting up and debugging this type of scheduled automation is to make sure your SAS program runs successfully *before* setting up the .cmd file and scheduled job.

WHAT WORKED WELL AND WHAT WE LEARNED

- Estimate your requirement collection timeframe to take twice as long as expected
- Make sure everyone knows their job and what is expected of them
- Assign owners and a Project Manager, again know what is expected
- Try to allow sufficient time for the build. There are usually time constraints but added pressure on the team makes the work more challenging
- Keep meticulous to-do lists and review details again and again
- More than one SAS programmer may be needed. It worked very well to have one main expert programmer and a separate programmer who tested and knew the needs of the business
- Don't have too many cooks in the kitchen. Stay lean if you can
- Build to the needs of business, build to be flexible, but also for efficiency and friendliness to the end user

CONCLUSION

Designing and building a custom campaign management system from scratch can be a practical solution to replacing an antiquated data management and application development environment. With the right recipe of a good team, time for proper requirements gathering, careful and smart system design, and efficient, friendly SAS code, it can be successfully done.

REFERENCES

- SAS Institute Inc. 2007. SAS Tech Note 648. "Example of Batch Processing Under Windows". Cary, NC SAS Institute Inc. Available at support.sas.com/techsup/technote/ts648/ts648.pdf
- Hurley, George J. 2006. "Xamining the X Statement (and Some Other Xciting Code)". *Proceedings of the SAS Users Group International 31 Conference.* Available at www2.sas.com/proceedings/sugi31/036-31.pdf

ACKNOWLEDGEMENTS

The author would like to thank Susan Bakken, Doua Xiong and Steve Hornbacher for working together and believing we could build this system and Tom Segar for proofreading this paper and providing positive support and smart feedback along the way.

CONTACT INFORMATION

Comments and questions are encouraged. Contact the author at:

Tera Olson
Aimia Proprietary Loyalty U.S. Inc., Minneapolis, MN
763-445-3584 (W)
763-350-0761 (M)
tera.olson@aimia.com