A Guide to Managing a SAS® Data Warehouse for Use Across an Intranet

Charles W. Bininger, Trilogy Consulting Corporation, Kalamazoo, Michigan

ABSTRACT

With the growing popularity of the Internet, companies have recognized that Intranets can be a valuable tool for dispersing data to their employees. SAS Institute has provided new tools for allowing Java™ applications access to SAS data sets. In late 1996, SAS Institute released a beta version of the Java Database Connectivity (JDBC™) driver that when used with SAS Share*Net® gave users the power to extract data to their desktop. While this paper will not discuss the implementation of the JDBC driver, it will provide direction and discuss issues to keep in mind when implementing the SAS Share*Net Server.

INTRODUCTION

As companies collect more and more information about their business, the need to organize, analyze, and distribute the data has become extremely important. Not long ago managers and analysts were forced to request information from programmers who in turn wrote programs to extract and manipulate existing data. The information was then forwarded to the requester in the form of reports. With any luck, the report satisfied the request.

When SAS developed Frame and SCL, it became easier for managers and analysts to access data. Developers could now create applications that allowed users to analyze and create reports on their own. This reduced the turn around time of acquiring data immensely. Although this was an improvement, it required additional software for most users (e.g. SAS, Emulation packages).

Today, with the increasing number of companies making use of Intranets for communication and information, it makes sense to use this evolving resource to distribute data to those who need it.

SAS SHARE*NET

In order to make data available to Java applets, the data must be managed by a SAS Share*Net server. In a perfect world, one could just define the existing data warehouse to the Share*Net server and everything should magically work. This is hardly the case.

Simply, SAS Share*Net is nothing more than a SAS session that listens through a port for data requests. The port is assigned in the operating systems services file. Once the server is started all options assigned to that session remains static. Here lies the problem!

Most established SAS data warehouses use formats within the data. This alone will not be a problem because a format search option can be defined before the actual Share*Net server is started. Since the requester of data is not a SAS session, all data with associated formats will be delivered as a formatted value. This is very convenient because it relieves the Java applet from concerning itself with cryptic unformatted values. So what is the problem?

THE PROBLEM

The problem is that most data warehouses do not use just one format library. A data warehouse may contain years of information in many SAS data sets. Consider the following scenario.

- The variable EDUC resides in data set xxx.xxx with a format of $EDUC.
- $EDUC is defined as follows…

```
PROC FORMAT LIBRARY=XXX;
VALUE $EDUC
  'A' = 'Some High School'
  'B' = 'High School Grad'
..'C' = 'Some College'
  'D' = 'College Graduate';
```

- The variable EDUC resides in data set yyy.yyy with a format of $EDUC.
- $EDUC is defined as follows…

```
PROC FORMAT LIBRARY=YYY;
VALUE $EDUC
  'A' = 'Middle School'
  'B' = 'High School'
  'C' = 'College Graduate';
```

- `OPTION FMTSEARCH = (XXX);`

If the variable EDUC is requested from data set yyy.yyy, the returned values from the Share*Net server will actually be incorrect. This happens because the FMTSEARCH option cannot be changed after the server is started.

POSSIBLE SOLUTIONS

Of course one possible solution is to rebuild all the data contained in the data warehouse. Formats would need to be renamed so that variables do not share formats that are defined differently. Another solution would be to actually change the values contained by the variable so that they could share formats. Both of these solutions would require a lot of time and work. There would also be no guarantee that existing programs would continue to work with these new architectural changes. There must be a better solution.

MULTIPLE SERVERS

The key to making the entire data warehouse available to Java without rebuilding the data is to have multiple SAS Share*Net servers managing the data. By separating data sets into groups defined by their format catalog, there will be no conflict of formats used by the data. However, this does introduce new problems.

- How do users or Java applets know which SAS Share*Net server to query for the data?
- How can I manage all these servers with a minimal amount of effort?

Again the solution is simple.

CREATE A MASTER SERVER

The function of the Master Server is to point the user or Java applet to the correct SAS Share*Net Server. This server can also manage all data sets that do not use formats and all data sets that use the most common format catalog. Start by creating a new SAS data set. Let us call it METADATA.LOCATION. This data set will contain the following variables;

- SERVER - Name of the Share*Net server.
- PORT - Port Number associated with the server.
- LIBREF - SAS libref
- DATANAME - 8 character name for the data.
- PATH - Physical name of libref.
- FMTCAT - Two level name of format catalog.

By querying the master server first, the user or Java applet can determine which server is

managing the desired data set. "METADATA.LOCATION" also serves another purpose. It can be used to dynamically create and manage all the needed servers.

Let us assume a UNIX environment.

1. Determine all the servers needed and add them to the services file.
2. Create an executable file as follows.
   - The first line in start_master_server starts the Master server.
   - The second line starts another SAS session and runs allocate.sas., which dynamically starts all the other servers.

start_master_server

```
/usr/local/sas612/sas
  -dmsbatch
  -fsdevice ascii.vt220
  -autoexec master_server.sas &

/usr/local/sas612/sas
  -autoexec allocate.sas
```

master_server.sas

```
libname metadata '<path>;
libname library  '<path fmt_lib>;
proc server server=master;
run;
```

allocate.sas

```
options
  source
  macrogen
  symbolgen;

libname metadata server=master;

/* Start all the servers in
METADATA.LOCATION except MASTER
which was started earlier */

proc sort
  data=metadata.location
  (where=(server ne 'MASTER'))
  out=work.server
  nodupkey;
  by server;
run;

/* Create autoexecs that start
each server and execute SAS in
the background using the
autoexecs */

data _null_;
  set work.server;
  call execute(
    'data _null_; file "' ||
    trim(left(server)) ||
    '.txt";');
  call execute(
    'put @1
      "options fmtsearch=(' ||
      trim(left(fmtcat)) ||
      ');";');
  call execute(
    'put @1
      "proc server server=' ||
      trim(left(server)) ||
      '; run;";');
  call execute(
    'call system("sas -dmsbatch
    -fsdevice ascii.vt220
    -autoexec ' ||
    trim(left(server)) ||
    '.txt");');
  call execute('run;');
run;

proc delete data=work.server;
run;

/* Allocate all the libraries to
the appropriate server */

proc sort
  data=metadata.location
```

```
  out=work.survey
  nodupkey;
  by server libref;
run;

data _null_;
  set work.survey;
  call execute(
    'proc operate server='||
    trim(left(server)) ||
    '; allocate library ' ||
    trim(left(libref)) || ' '||
    trim(left(path)) ||
    '; run;');
run;

/* Delete all the temporary
autoexecs created earlier */

data _null_;
  call system('rm *.txt');
run;

/*Display all servers started and
libraries allocated to each
server */

proc sort data = work.survey
  nodupkey;
  by server;
run;

data _null_;
  set work.survey;
  call execute (
    'proc operate server=' ||
    trim (left (server)) ||
    ' display library _all_;' ||
    ' run;');
run;

proc delete data=work.survey;
run;
```

IMPLEMENTATION

All that remains to take advantage of the data warehouse is to define the procedure for accessing the data.

- Make all initial queries to the master server to find which server is managing what data. Make sure to ask for the port number.
- After determining which server has the data you want, a query can now be sent to the appropriate server.
- If an unformatted value is needed from a data set use a PUT function in your SQL select statement. All SAS functions and expressions are available for use. This is especially handy for dates and manipulating returned data from the server.

CONCLUSION

The rapid growth of the Internet has produced emerging technologies that greatly influence how data is distributed in the business environment. Answering this current trend, SAS has provided the tools that allow us to take advantage of the Internet. However with any new technology, one must be careful to implement these tools correctly with current data warehouses. Had we blindly taken advantage of the JDBC driver and SAS Share*Net, inaccurate data may have been passed on to the user. Even worse, business cases and decisions would have been based on this data.

REFERENCES

SAS Institute, Inc., *SAS/Share Software: Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1991

Hoyle, Larry (1997) "Choosing a Method for Connecting Java to SAS System Across the Internet- CGI, JDBC, or Socket," in *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

Walters, Barbara (1997) "Exploiting Java Technology with the SAS Software," in *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*. Cary, NC: SAS Institute, Inc.

http://www.sas.com/md/web - Use this URL to download a beta of the SAS/SHARE*NET driver for JDBC. SAS Institute provides sample code with the SAS/SHARE*NET driver for JDBC.

ACKOWLEDGEMENTS

Special thanks to …

Joseph Turckes	Ford Motor Co.
Glen Keer	CPI
Mark Aretha	SAS Institute
Chip Kelly	SAS Institute
Mark Kurowski	Sun Microsystems
Bill Kefalas	Sun Microsystems
John Bekas	NeoGlyphics Corp.
Jack Fuller	Trilogy Consulting
Kevin Brown	Trilogy Consulting

SAS and SAS/SHARE*NET are registered trademarks of the SAS Institute Inc. Java and JDBC are registered trademarks of Sun Microsystems, Inc.

® indicates USA registration.

The author may be contacted at:

Charles W. Bininger
Applications Developer
Trilogy Consulting Corporation
5278 Lovers Lane
Kalamazoo, MI 49002
(616)344-4100 voice
(616)344-6849 fax
Internet: cwbining@trilogyusa.com