Using SAS/CONNECT® to Control a Distributed Application

Roger Williams
Dimensional Insight, Inc., Broomall, PA

Abstract

As Windows NT™ servers become more powerful and commonplace, a departmental installation of the SAS® System can easily be configured to combine the flexibility and resources of the desktop (Excel, Word, etc.) with the power of the Windows NT server (centralized data storage, faster processor and specialized software). This paper lays out the steps required to set-up such a departmental configuration of the SAS System including SAS/CONNECT, discusses the code required to enable it, and demonstrates the implementation from within SAS Display Manager. The SAS program will execute on the local host and the remote host, launch a *remote* DOS application to access the SAS generated data, and then launch a *local* Windows application to view the results of the remote processing.

Background

Dimensional Insight, Inc. (DI) provides unique multi-dimensional data analysis and reporting solutions enabling you to rapidly explore data, gaining knowledge and making decisions with unprecedented speed, flexibility, efficiency and accuracy.

DI-Diver™ provides easy "*point and click*" access to data stored in a file called a DI model. This single-file DI model contains unique, proprietary data structures that simulate multi-dimensional characteristics. The model itself is created by feeding data to a DI build module (desktop DI-Diver or server-based DI-Atlantis™). All this results in quick, consistent response at "the speed of thought" (OLAP).

The Dive Macro was developed by DI to enable you to leverage your use of the SAS System to make them more productive by automating the data transformation process. The Dive Macro easily transforms any data contained in a SAS data set into the files required by the DI model build process. The DI model can then easily be built from within a SAS session or as a separate step or job. The Dive Macro is intended to run on any workstation running Windows 95/NT or any server supporting a DI-Atlantis build module (MVS, OS/400, UNIX, etc.) and the SAS System (version 6.11 or higher). The Dive Macro can be executed via display manager, command mode, or batch.

A design objective was to create a single macro to reduce maintenance and to make it available in a non-compiled form to allow individual sites to modify it to suit their environment. Hence, emphasis was placed on the software interface and not on the SAS System platform interface. Each site could address this if deemed appropriate.

Dive Macro Implementation

The Dive Macro in combination with the SAS System can be implemented on the desktop, on the server or in a SAS client/server environment. The desktop or workstation is defined as any PC which can execute both SAS and the DI products. This is generally a PC running Windows 95/NT.

Desktop or Workstation mode is intended to function as an extension to an interactive SAS session providing you with easy "*point and click*" access to their data (DI-Diver). All required file allocations are performed by the Dive Macro making it easy for you to get up and running. This is possible since the Windows 95/NT are very similar in both structure and available commands.

Server mode is intended to simplify the creation of the input files required for the DI model build process (it does not actually build the DI model or launch DI-Diver). This is accomplished by leveraging the use of the SAS System. Since DI-Diver does not execute on a "server," this mode is most useful for traditional "production-type" applications.

Client/server implementation of the Dive Macro is an interesting combination of the desktop and server implementations. This mode provides a way to use a remote Windows NT server as a compute engine to build DI models without the need for you to be physically sitting at the server (which is usually locked away in an IT secured location) and without installing terminal emulation software (lagging for Windows NT compute servers).

Exploiting SAS/CONNECT

The DI product family provides numerous implementation scenarios allowing a firm to leverage its investment in existing hardware and software or take advantage of economies of installing new computing platforms; most notably the Windows NT operating system running on relatively inexpensive INTEL-based hardware platforms. DI-Atlantis installed on such a platform provides excellent functionality at a moderate price and is ideal for a departmental installation. There is a problem however! The Windows NT server must be accessible as a batch compute server and not just as a file server. Enter SAS/CONNECT!

The compute services capabilities of SAS/CONNECT provide access not only to the files, but also the hardware resources of the remote machine. Large volumes of data can be accessed and manipulated on the server while the desktop client handles the GUI thereby minimizing network traffic. The alternative calls for you to replicate the data across many desktops resulting in increased network traffic. The need to refresh the data only exacerbates the problem. This all results in needless delays for each analyst.

In this environment the SAS/CONNECT RSUBMIT (remote submit) statement can be used to access and manipulate data as well as to execute the Dive Macro to create the required files to build a DI model. The X statement can then initiate the DI-Atlantis model build process. The resulting DI model can then be easily accessed using DI-Diver.

This paper focuses on the issues faced in implementing the Dive Macro in a Windows NT client/server environment using SAS/CONNECT.

The Demonstration Environment

The assumption is that departmental data is located on (or accessible from) an Windows NT server (SAS software®, ORACLE™, DB2™, SYBASE™, etc.) where SAS software (data access/manipulation) and DI-Atlantis (DI model build

module) are also installed. Further, SAS software (program libraries/output viewing) and DI-Diver (desktop visualization and reporting) are installed on a Windows NT workstation. The workstation and the server are networked together using TCP.

To demonstrate the Dive Macro interface, SAS software and DI products have been installed onto a laptop computer running Windows NT Workstation 4.0. The software consists of base SAS software (v6.12), SAS/CONNECT, DI-Diver, DI-Atlantis and the Dive macro. The laptop computer will function as both the client and the server! This obviously is not a normal installation but is identical to the more traditional client/server installation within a workgroup or enterprise.

How SAS/CONNECT Works

To use SAS/CONNECT, the SAS System and SAS/CONNECT must be installed on both the local and remote hosts; in this case the client workstation and the Windows NT server. Obviously some type of network is in place (e.g., Microsoft Lan Manager) that allows the local host to initiate a link with the remote host. This link is usually established automatically at the time you initiate a local SAS session (although it can be initiated manually from within display manager). Each time a link is established a SAS session is invoked on the remote host. SAS/CONNECT statements and procedures are used to communicate between hosts. The SAS log and output produced by the remote SAS session are directed back to and displayed on the local host.

The link is established by executing the SAS/CONNECT SIGNON command which executes a series of special SAS statements contained in a file called a script. Several sample scripts are provided as part of the SAS/CONNECT installation. The script is operating system and communication protocol dependent.

When accessing a remote Windows NT host the SAS Institute supplied SPAWNER program must be executing on the remote host. This program, installed as a Windows NT service, listens for incoming requests to connect the local and remote SAS sessions. New in v6.12, the SPAWNER program running on Windows NT using the TCP/IP access method provides for access security and communication port access restrictions. These options are not applicable on a laptop configuration.

Getting Up and Running

The first step is to install the SAS System and SAS/CONNECT on both the local and remote host. On our laptop configuration both the local and remote hosts will share the SAS software. Next the SPAWNER program must be installed on the remote host as a Windows NT service using the TCP access method. At a DOS prompt :

```
c:\sas\connect\sasexe>  spawner  -install  -comamid tcp
c:\sas\connect\sasexe>  net start "sas job spawner"
```

The installation of SAS/CONNECT is well documented by SAS Institute and should problems arise at your site SAS Institute's Technical Support is just a phone call away. The online help for v6.12 is an excellent source for additional information for all aspects of SAS/CONNECT and the SPAWNER program in particular.

Once the two systems are operational the local host needs to sign onto the remote host. As previously mentioned it is very common to do this as part of the start-up procedure for the local SAS session. The -AUTOEXEC option allows you to specify a SAS program to execute at start-up. This option is specified in the local host **config.sas** file as :

```
-autoexec "c:\di-demos\di-desktop.sas"
```

The SAS program **di-desktop.sas** simply references a *script* file, specifies communication access method options and issues the SAS/CONNECT SIGNON statement :

```
filename Atlantis 'c:\program files\sas\di-server.scr';
options  comamid=tcp  remote=rogernt ;
signon Atlantis;
```

The TCPWIN.SCR script (part of SAS/CONNECT installation) was modified slightly and renamed **di-server.scr**. As part of the remote signon a SAS session is invoked using SAS script statements :

```
type  'sas  -dmr' ;
type  '     -comamid   tcp' ;
type  '     -autoexec   di-server.sas'   LF ;
```

The DMR option invokes a remote version of the display manager; the COMAMID option identifies the communication access method and the AUTOEXEC option specifies a SAS program to execute each time the SAS session is invoked. In this case the **di-server.sas** program simply invokes the SASAUTOS option (where to look for SAS macros; in this case the Dive Macro) :

```
options sasautos=("c:\DiveMacro"   sasautos        );
```

At this point the installation is ready to be "tested." When the SAS System is executed on the local host the di-**desktop.sas** program is executed which automatically signs onto the remote host using the **di-server.scr** script that executes the SAS System, which automatically defines SAS macro libraries (**di-server.sas**). To insure that everything is working the following program can be executed.

```
%dive;                          /* local host  */

RSUBMIT ATLANTIS;
%dive;                          /* remote host */
ENDRSUBMIT;

%dive;                          /* local host  */
```

The first and last "%dive;" are executed on the local host resulting in "ERROR 180-322: Statement is not valid or it is used out of proper order." Simply; it can not find the Dive macro on the local host. The RSUBMIT statement followed by the *remote-session-id* (filename reference to signon script - Atlantis) causes all SAS program statements following it and preceding the ENDRSUBMIT statement to execute on the remote host.

All SAS statements to be executed on the remote host must be written as if you were sitting at the remote host. For example, the remote host may be connected to the local host as a file server and be referenced as drive "g:" but if it is actually the "c:" drive on the remote host then that is how you should reference it. Keep this "relative" vs "absolute" difference in mind as you construct SAS programs.

At this point, client/server applications can be written and deployed within an organization. It can be this simple!

The Dive Macro - an Overview

By specifying options much like you would with a SAS procedure, a copy of the data contained in user-specified SAS data set will be transformed into a DI model and presented to you for additional interactive analysis using DI-Diver. Every attempt is made to take full advantage of the work performed by the business analyst (using the SAS System) to define the SAS data set (i.e., labels and formats). All data contained in the model are as they would appear if a PROC PRINT were executed on the SAS data set. All SAS formats and user-created formats are supported.

The Dive Macro was created as a "keyword" parameters macro. For such a macro the "keyword" (the SAS macro variable name) is followed by an equals sign ("=") and the user supplied value. If a parameter and value are not specified the "default" value is assumed. The order of the "keyword" parameters is not important.

Dive Macro Example :

```
%dive(data   =   save.sales,
      class  =   customer  [name  city  state]
                 branch    [manager  region]
                 product
                 month     [quarter  year],
      var    =   units  sales  pln_dol  pln_unit);
```

The Dive Macro is "smart" enough to determine the hardware and available software and to do the "right" thing - "transform" the data, "build" if appropriate, and "dive" if appropriate. The DI model build process requires (1) **input data** - a tab-delimited file consisting of 1 observation per record, (2) **input data dictionary** - a description of the columns (and their order) contained in the input data, (3) **model description** - a categorization of columns (SAS variables) into dimensions, summary, or info fields, and (4) **user view** - default "diveplan" used when the model is initially opened and displayed to you. The Dive macro automates the creation of these files ("transform" function).

All file allocations are performed automatically by the Dive Macro when executed in workstation mode. By default the location is the SASWORK sub-directory created by the SAS System for that particular SAS session. Thus, when the SAS session is terminated all "temporary" files created by the Dive Macro are deleted.

Server or client/server mode requires that all files be pre-allocated prior to executing the Dive Macro. This is due to differences in operating systems. The five (5) files which need to be allocated are :

Input text file to contain the tab-delimited values of the SAS variables to be in DI model

DI extension :	*.TXT*
Filename :	*INPUT*
lrecl :	*10000*
recfm :	*v*

Dictionary file to contain the order of the SAS variables in the Input text file.

DI extension :	*.DIC*
Filename :	*DICT*
lrecl :	80
recfm :	*fb*

Description file to contain the description of the model to be built

DI extension :	*.DSC*
Filename :	*DESC*
lrecl :	80
recfm :	*fb*

Diveplan file to contain the description of the appearance of the DI-Diver window

DI extension :	*.DVP*
Filename :	*DVP*
lrecl :	80
recfm :	*fb*

Model file to contain the DI model.

DI extension :	*.MDL*
Filename :	*MODEL*
lrecl :	80
recfm :	*fb*

Once the SAS data set has been "transformed" the DI-Atlantis build module can be executed using the X statement :

```
X  builder.exe  sales.dsc;
```

Every product has their own unique terms and the DI products are no different. Fortunately DI and the SAS System have a lot in common. Using DI-Diver is akin to "diving" into the output data set created by PROC SUMMARY or PROC MEANS. A listing of key DI terms related to the SAS System follow :

- **dimension (core) -** similar to SAS variables named in CLASS statement of PROC SUMMARY or PROC MEANS. Numeric or character SAS variables may be specified. Maximum of twelve core dimensions.
- **summary -** similar to SAS variables named in VAR statement. Variables for which summary stats are to be calculated (e.g., sum, min, max, std).
- **info -** similar to SAS variables contained in an ID statement. Each value of an info field provides additional information about a dimension. Each core dimension may have multiple info fields associated with it. A given dimension value can have only a single value of an info field associated with it.
- **dimension (dynamic) -** an info column can be transformed into a dynamic dimension and assume the characteristics of a core dimension.

The Final Application

All the pieces are now in place; the SAS System and SAS/CONNECT are installed on both the local and remote hosts, the SPAWNER program is executing as a Windows NT service on the remote host, the DI-Atlantis build module can be executed on the remote host and DI-Diver can be executed on the local host. Now let's put it altogether :

```
RSUBMIT ATLANTIS;

filename formats "C:\DI-Demos\ub-92\fmtlib.sas";
%include formats;

libname  save  "c:\DI-Demos\ub-92";

filename input "c:\DI-Demos\temp\ub-92.txt"  lrecl=10000;
filename dict  "c:\DI-Demos\temp\ub-92.dic"  lrecl=80;
filename desc  "c:\DI-Demos\temp\ub-92.dsc"  lrecl=80;
filename dvp   "c:\DI-Demos\temp\ub-92.dvp"  lrecl=80;
filename model "c:\DI-Demos\temp\ub-92.mdl"  lrecl=80;

%dive  (data   =  save.sample2,
        class  =  s_t_d_p   [asource  atype    disp
                             pay1                    ]
```

```
                drg10     [mdc10]
                dcchpr1
                pcchpr1
                ag_sx     [age      sex]
                dates     [adaywk   amonth   aqtr]
                hospital  [hospaddr hospcity hospst
                           hospzip  h_bedsz  h_contrl
                           h_loc    h_tch    totdschg ],
         var  =  died
                 los
                 totchg
                 prday1,
        name  =  UB-92                              );

options xwait xsync;
x  builder  c:\DI-Demos\temp\ub-92.dsc;

ENDRSUBMIT;


options noxwait noxsync;
x  DI-Diver c:\DI-Demos\temp\ub-92.dvp;
```

Coder's Corner

While developing this application several SAS System features were used for the first time and many familiar features were re-visited. I would like to share some of these with you now. If you have more efficient ways to accomplish any of these tasks I would appreciate your feedback.

Problem: Treat the Dive Macro as if it were a SAS procedure and provide notes and error messages you, the SAS user, are already familiar with. This includes the execution time.

Solution: When NOTE: or ERROR: are printed to the SAS log beginning in the first column The SAS System will interpret these as though it produced them and enable overprinting and color coding (blue for notes; red for errors, etc.).

```
%let _beg   = %sysfunc(datetime(),15.2);

data _null_;
  call symput("_time",compress(put(datetime() -
&_beg,7.2)));

data _null_;
  file log;
  put "NOTE: The DIVE MACRO executed in &_time seconds." ;
```

Problem: The INPUT file is a tab-delimited file. The tab character differs between and ASCII system such as a PC and an EBCIDIC system such as MVS.

Solution: These is no SAS parameter to detect the system type. The $HEX4. format can be used to determine if the "ll" converts to "6C6C". If it does we know it is an ASCII system and "09"X should be used; else it is an EBCIDIC system and "05"X should be used.

```
%if %sysfunc(compress(ll),$hex4.) = 6C6C %then
                                    %let _delimit = "09"X;
%else                               %let _delimit = "05"X;
```

Problem: DI programs need to be executed from within the Dive Macro. If not in the system path then one needs to be able to find them. During the DI-Diver setup.exe install program a diver.ini file is created containing the DI program locations. If we can find and read this file we will know where to find the DI programs. This location can vary among systems (e.g., Windows 95/NT, etc.).

Solution: The host environment variable WINDIR contains the complete path for the locations of all .ini files. The %SYSGET function obtains values for host environment variables.

```
%if %sysfunc(fileexist(%sysget(windir)\diver.ini))…
```

To view all host such variables open a DOS window and issue the DOS SET command or execute the following program. All variables will be listed.

```
Filename foo pipe "set";
data _null_;
  infile foo;
  input;
  put _infile_;
```

Problem: Once the diver.ini is located and read it is now necessary to determine if the di-diver.exe can be found in the specified location.

Solution: In non-macro code the FILEEXIST function could be used but this is not a macro function. However, the %SYSFUNC macro function can be used to evaluate any non-macro code function within the macro processor. This is a relatively new feature.

```
%if %sysfunc(fileexist("c:\di-diver.exe")) = 1 %then %do;
```

An alternate solution is to use the system return code for a FILENAME allocation - &SYSFILRC - to determine if a file exists. A value of 1 indicates a successful allocation.

```
filename _di "c:\di-diver.exe";
%if &sysfilrc ne 0 %then %do;
  %let _error = 9;
  %goto _error;
%end;
```

Problem: It was a design objective to create a single macro which could execute on multiple platforms. Any system which can execute DI-Diver is defined as a workstation and the macro acts accordingly. If this workstation is also being used as a remote compute engine fine then the Dive Macro needs to treat the system as a server.

Solution: The remote host in a SAS/CONNECT implementation must start the SAS System with the DMR option set. This option invokes a remote version of the display manager. The local host has a value of NODMR for this option. The GETOPTION function used in combination with the macro %SYSFUNC function can obtain the setting for any SAS option.

```
%if %sysfunc(getoption(DMR)) = DMR …
```

Problem: When executed on a workstation the Dive Macro allocates all files. When executed on a server files are allocated by the user. To handle both situations the file names need to be assigned to macro variables.

Solution: The PROC SQL view (VEXTFL) for the table DICTIONARY.EXTFILES lists external files associated with a current SAS session by filerefs. Since specific filerefs must be used in conjunction with the Dive Macro the complete path for these external files can be obtained. For more information refer to SAS® Technical Report P-222 SAS® Software Changes and Enhancments. - Dictionary-tables.

```
data _null_;
  set sashelp.vextfl;
  where  fileref in ("INPUT","DICT","DESC","DVP","MODEL");
  call symput("_" ||fileref,trim(upcase(xpath)));
```

Problem: When executed on a workstation the Dive Macro creates temporary files (batch files, transformation files, etc.) which need to be "transparent" to the user and then "cleaned-up."

Solution: The SAS System creates the SASWORK library as its temporary work space. Since it is just a directory why not use it - when the SAS session terminates normally the directory is deleted! The PROC SQL view (VMEMBER) for the table DICTIONARY.MEMBERS provides information about SAS data libraries associated with a current SAS session by librefs. For more information refer to Technical Report P-222. - Dictionary-tables.

```
data _null_;
  set sashelp.vmember;
  where libname = "WORK";
  call symput("_work",trim(path));
```

Problem: When viewing information it is better to see descriptions rather codes and dates rather than numbers. You should not be required to key this data multiple times simply because another software package is being utilized. It would be nice to be as "transparent" as possible.

Solution: SAS meta data (labels, formats, etc.) is stored as part of each SAS data set. Access this information and create macro variables for each variable containing an attribute.

```
proc contents data=sales
               out=_temp (keep   = name     label  format
                                   formatd  type   formatl)
              noprint ;

data _null_;
  set _temp;
  call symput(trim(name),trim(label));
```

Problem: As part of the error-checking routine it is necessary to terminate the Dive Macro as soon as an error is detected. One such situation occurs when your-specified data set contains zero observations, probably due to an error elsewhere in your program.

Solution: Create a macro variable containing the number of observations in your specified data set and perform conditional %if logic.

```
data _null_;
  if 0 then set sales nobs=count;
  call symput("_obs",count);
  stop;
run;

%if &_obs = 0 %then %do;
  %let error = 6;
  %goto _error;
%end;
```

Problem: When executed on a workstation the appropriate DI model build module and DI-Diver need to be executed. This requires that files be deleted if necessary and directories need to be changed.

Solution: The X statement can be used to pass a command to the system. On a workstation this results in the opening of a DOS window, execution of the command and the closure of the DOS window. There is no way to execute DOS commands as "background" tasks. For several commands this means DOS windows will repeatedly be opening and closing. Not very visually appealing. To open a single DOS window and execute multiple commands place all commands in a batch file.

```
data _null_;
  file "c:\sas\saswork\#td74203\temp.bat";
  put "echo off"          /
      "c:"                /
      "cd programs"       /
      "di-diver.exe  c:\sas\saswork\#td74203\sales.mdl"  ;

x  c:\sas\saswork\#td74203\temp.bat;
```

Problem: When executed on a workstation using the DI-Atlantis build module, a command window opens to execute the build module followed by a second command window to launch DI-Diver to view the model just created. DI-Diver should be launched only after the model has been created. All this should happen automatically without your intervention.

Solution: Specify the SAS options NOXWAIT and XSYNC. The command prompt window will close automatically (NOXWAIT) and the SAS System will wait for the application to finish (XSYNC).

```
options  noxwait  xsync;
x  builder.exe  sales.dsc;
```

Problem: When DI-Diver is launched to view the DI model you should be able to toggle between the SAS session and DI-Diver.

Solution: Specify the SAS options NOXWAIT and NOXSYNC. The command prompt window will close automatically (NOXWAIT) and the SAS System does not wait for DI-Diver to finish (NOXSYNC).

```
options  noxwait  noxsync;
x  di-diver.exe  sales.dvp;
```

Problem: The macro variables created during the Dive Macro should not impact the user environment.

Solution: Define macro variables created inside the macro as local macro variables.

```
%local  data      class     var       missing   where
        obs       labels    name      comments  footer
        ignore    build     block     password  stats
        model     dive      builder                 ;
```

Problem: This is a complex macro (multiple platforms executing multiple products) which may require code changes and enhancements over time which will require quick understanding of the code flow.

Solution: Document! Document! Document! I prefer to box comments using the /* */ comment style. It is

visually clean and crisp and can include semi-colons. I also find it useful to comment sections of code (multiple data and proc steps) so as not to break-up the SAS code too much. I also recall reading (source unknown) that these comments are not compiled should you need to compile SAS code.

```
/*-----------------------------------------------------------*\
| create macro variables from info found in diver.ini file  |
|-----------------------------------------------------------|
| 1. Read diver.ini file for locations of builder and       |
|    diver [Diver Path, Builder Path]; create macro vars    |
| 2. check to see if paths and files exist.                 |
\*-----------------------------------------------------------*/
```

Problem: Was the SAS code successfully executed on the remote host ?

Solution: Use the %SYSRPUT function on the remote host to create a macro variable which can be evaluated on the local host.

```
%macro dive (……..);
    %sysrput _error = &_error;
%mend dive;

rsubmit DI-Atlantis;
    %dive (……..);
endrsubmit;

%macro local;
    %if &_error ne 0 %then %goto halt;

    %halt:
%mend local;
%local;
```

Conclusion

The compute services capability of SAS/CONNECT allows you to take full advantage of the power and economies of an NT server. In addition, network traffic can be reduced by processing large SAS data sets on the remote compute server. Specialized software can also be accessed by multiple users resulting in a more effective and efficient information system.

Acknowledgments

Author Contact

Roger Williams
Dimensional Insight, Inc.
2193 West Chest Pike
Broomall, PA 19008

610-359-8401
roger@dimins.com
http://www.dimins.com/sas