

SAS Application Development Using Windows RAD Software for Front End

Zhuan (John) Xu

Blue Cross Blue Shield of Iowa & Big Creek Software, Des Moines, IA

Abstract

This paper presents a method for using SAS as a data management system and analytic engine, with a front end implemented with a Windows 95/NT development tool such as Borland Delphi or Microsoft Visual Basic. Two approaches are introduced for connecting the SAS engine and the front end. One uses the FTP protocol to implement a client/server application, where the server can be any computer with an FTP server and SAS installed, such as a UNIX workstation. The other uses OLE Automation to control SAS on Windows 95/NT. This presentation discusses how to develop such applications. One client/server application using FTP is presented as an example. In addition, a Windows application which uses SAS through OLE automation is provided free to the public as a front end to SAS.

Introduction

At the Medicare division of Blue Cross and Blue Shield of Iowa, SAS has been used to analyze insurance claim history data for non-covered services, over- or under-utilization, abuse, and practice patterns. It is also used in Small Area Analysis and other more advanced studies in health services. SAS was selected for its power in data manipulation and statistical analysis.

In the early stage, SAS programs were developed and reports were generated for users. This approach was modeled after the mainframe environment, where users typically work with pre designed reports. As users often need direct access to data to perform analysis themselves, we tried various methods to provide such functionality. These included using Microsoft Excel with data or analytic results from SAS; using Microsoft Access applications with data extracted from a SAS data warehouse; and SAS applications developed with SAS AF/Frame. Through years of experience, we observed that users work best with a well-designed graphical user interface, consistent with one they use daily and thus are familiar with. This means Windows 95 for most people. As a result, we developed MAuditor, a decision support systems with SAS in the back end and a PC database application developed with Borland Delphi at the front end. These two sides are connected together using the FTP protocol. To the user, it is an integrated application.

Another way to develop SAS applications is through OLE automation. Since Version 6.11, SAS can be used as an OLE automation server on Windows 95/NT. OLE automation can be used easily to develop a local application. It can also be used in client/server

development. Recently, Microsoft released DCOM on the Windows NT and 95 platforms. DCOM allows OLE automation controllers and servers to reside on different machines. This significantly simplifies client/server development.

In this paper, both approaches will be discussed in more detail, with two example applications. The first one is MAuditor mentioned above. The second one, EasySAS, is a demonstration application developed with Delphi using SAS as an OLE Automation server. EasySAS is available free for download at <http://www.bigcreek.com/sas>.

MAuditor – A FTP Based Decision Support System

MAuditor is a decision support system implemented to provide help in developing, applying, monitoring, and revising medical policies used by Blue Cross Blue Shield of Iowa to regulate Medicare services. Based on analysis performed using MAuditor, we may educate providers, request refunds, stop improper payments, or in some cases, provide certain services to reduce or eliminate future costs to the Medicare program.

The Structure of the MAuditor System

Figure 1 illustrates the hardware architecture of the MAuditor. An HP UNIX workstation is used as the database server. It hosts a SAS data warehouse and performs analysis as well. It is attached to a local network running Netware with support for the TCP/IP protocol. All PCs on the network can access the HP workstation through the workstation's built-in FTP server. The Windows front end of MAuditor can be installed on any PC running Windows 95 or NT. In addition, the front end is linked to the documentation of Medical Policies stored on the network file server so users can view selected policies within MAuditor. It should be mentioned that there is nothing special about the HP workstation. Another UNIX computer, Windows NT Server or workstation can also be used, as long as SAS and an FTP server is available on the computer.

Figure 2 shows the software components on the server. There are three major components:

- **Database:** MAuditor uses SAS as a database server. SAS macro programs and UNIX scripts were developed to load and clean the data. The resulting SAS datasets were sorted for better performance. A special clustered index is used to further speed up data

extraction. Various look up tables, containing data on medical procedures, diagnoses, providers, demographics etc. are used in MAuditor. The claim history data are stored in several datasets, based on time and type of services. They are automatically concatenated in a macro program. Currently, there are over 35 million observations used in MAuditor, which is about 50% of the data warehouse. Sorting and indexing have a very big impact on the performance. Without them, it took about 40 minutes to just extract data since SAS perform sequential reading. It is down to 1-2 minutes with the clustered index now. The clustered index used in MAuditor is similar to the one developed by Daniel Dower. For details, please refer to his SUGI 20 paper 'Access Data Faster than a SAS Index'. A SAS utility program is used to convert SAS datasets to dBASE format which can be transferred to PC and used by MAuditor or other applications.

- **Analytic component:** The biggest advantage of building data warehousing in SAS is SAS's excellent analytic procedures and data manipulation capacity. Naturally, MAuditor uses SAS macro programs to do various analyses. After history data are extracted from the data warehouse, it is used to perform analysis on medical procedures and patient diagnoses. Providers are ranked in terms of non-covered services. In addition, trending analysis is conducted to check effectiveness of the policy and any measures used by Medicare to correct problems. The result is in a standard SAS output file.
- **The communication component** is pretty simple yet is the key to MAuditor. On the server side, a UNIX script was written to act as a server. Its primary function is to run a SAS program when it receives a request from a client PC. Its function can be described in the following flow chart:

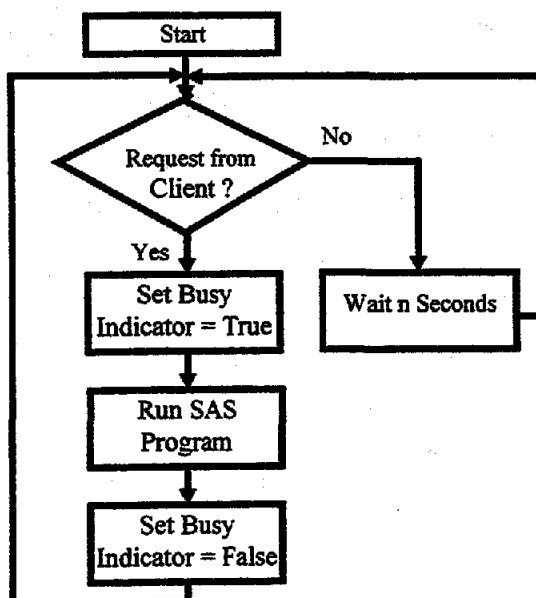
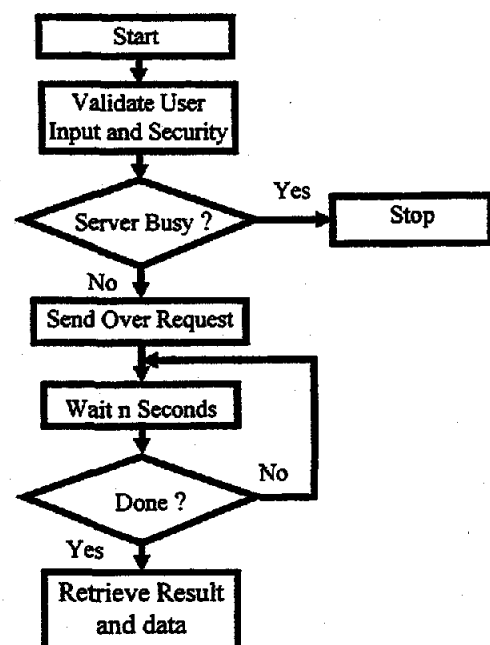


Figure 3 shows software components on the client side.

- **Medical policy database:** The front end itself is a standalone medical policy database application. It displays medical policies in term of medical procedure codes (CPT) and covered diagnosis codes (See Figure 4). Users may view a complete policy for more details. It also allows users to search policies by CPT or diagnosis codes, or by their descriptions (See Figure 5). Users with proper security may revise policies.
- **Dynamic query tool:** The dynamic query tool is provided for further analysis. Users can use the query tool to subset, sort, or summarize data in many different ways (See Figure 5). For data with a description, such as procedure code, double clicking on the data cell will show the matching description.
- **Report/Chart:** The report and chart is created in MS Excel through OLE Automation. This significantly reduced development time and allows users to fine tune their reports and graphics.
- **Mapping:** A data map of Iowa counties can be generated for Small Area Analysis. Other data at county level or at ZIP code level can also be added to the map.
- **Remote function:** The query tool, report, chart, and map mentioned above are dependent on the result of analysis or data from the server. Therefore the key is to set up analysis parameters and then send requests to server. Figure 5 shows the set up screen. The client communicates with the server through the FTP protocol. The following flow chart is the logic used in the front end:



EasySAS – A Utility Program

EasySAS is a Windows utility program developed mainly as a demonstration of controlling SAS through OLE Automation. It provides a simple graphical user interface to basic SAS functions so users can use SAS without writing a single line of code. Its import function makes it much easier to get data from text files and Excel spreadsheets into SAS. Similarly, its export function can export a SAS dataset to a text file or Excel file.

Easy SAS was written using Borland Delphi.

SAS as OLE Automation Server

OLE Automation is a mechanism through which a Windows application can control another application programmatically. This technology is currently widely used in Windows 95/NT. For example, all applications in Microsoft Office can be used as OLE automation servers and clients. Most Windows application development tools, such as Microsoft Visual C++, Visual Basic (VB), and Borland Delphi support OLE Automation. Since version 6.11 SAS can perform as an OLE Automation server. Through OLE Automation, SAS provides another application with a programmable object. This means that you can use an application that can act as an OLE automation controller to create a SAS session and control it using the methods and properties that the SAS System makes available.

Currently, SAS only exposes a few properties and methods. Even so, it enable us to develop applications that are fully functional. To invoke a SAS session in VB, we need:

```
Dim OleSAS as Object
Set OleSAS = CreateObject("SAS.Application")
```

It is very similar in Borland Delphi:

```
OleSAS : Variant;
OleSAS := CreateOleObject('SAS.Application');
```

After SAS is started we can submit SAS programs. The following example in VB executes a simple DATA step:

```
OleSAS.Submit("data _null_ ; x=1; run;")
```

It is almost identical in Delphi:

```
OleSAS.Submit('data _null_ ; x=1; run;');
```

To run the SAS program `c:\test.sas`, we can replace the SAS code in quotes with `%include "c:\test.sas"`. This is very much like running SAS program in batch mode. Now, to create an application, all we have left to do is to generate the right SAS program, which is similar to the dynamic SQL commonly used in client/server development.

Other properties and methods are documented in SAS online help file.

The Structure of EasySAS

The main screen of EasySAS is a file manger shown in Figure 6. SAS is invoked through OLE automation at the same time EasySAS is started and stays invisible all the time. The user then clicks on the search button to get a list of directories containing SAS datasets in the left list box. The right list box shows SAS datasets in the selected directory. Double clicking on a SAS dataset will display the contents of the dataset and up to the first 20 observations in a built in editor. Basic file manger functions, such as copying, moving, deleting, and renaming files, and creating and deleting directories, is also implemented. In addition to SAS datasets, the user may select and display any SAS program. Other types of file filters can also be selected. In this case, double clicking on the file will open or run the file with associated application. Several SAS procedures are available in EasySAS. For example, Descriptive Statistics is an implementation of PROC MEANS. When invoked, a screen will show up (Figure 6), displaying the name of the SAS dataset, number of observation in the dataset, a list of variables, their type (Numerical, Character, or Date), and their labels if available. The user then can use buttons or 'drag and drop' to select the variable(s) to be analyzed and to group variable(s). A list of statistics is available for users to select. By default, Mean, Sum, and Standard Deviation are checked. Clicking on the OK button will generate a SAS program and submit it to SAS through OLE Automation. The result is displayed in the editor (Figure 6). The user can also view the log file and the generated SAS program. Furthermore, the user may modify the generated SAS program and resubmit it within editor.

Now let's take a look at the implementation of PROC MEANS in Easy SAS. First, it retrieves information on the SAS dataset. When the user selects Descriptive Statistics from menu, EasySAS runs the following SAS program:

```
proc printto print= log='c:\temp\_SASFM.log';
options linesize=72 pagesize=56 nodate
        pageno=1;
libname _DATAIN "c:\sas\core\sashelp";
proc contents data=_DATAIN.salary
        out=_templ noprint;
data _null_ ;
set _templ;
length T $ 1;
file 'c:\temp\_SASFM.lst';
if type=2 then t='C';
else if index(format,'YY' )>0
    or index(format,'DATE')>0
then t='D';
else t='N';
put name $8. ' ' t $1. ' ' label $ 30. ' '
    nobs 10.;
run;
proc printto;
run;
```

The output is in the text file c:\temp_SASFM.lst

BEGDATE	D		319
ENDDATE	D		319
IDNUM	N	Identification Number	319
JOBCODE	C		319
SALARY	N	Salary	319

Note that Column 1 is the variable name. Column 2 is the type of Variable, with D for Date, N for Numeric, C for Character. Column 3 is the variable label, and the last column is the total number of observations in the dataset.

This file is then read back by EasySAS and the data is used in the input screen (Figure 6).

In the input screen, the user sets up the analysis to be performed. It is necessary to validate user input. For example, at least one variable must be selected for analysis and it may not be a character variable. Input validation is often more difficult and time consuming to program than generating the SAS program itself. To minimize the required validation, EasySAS, whenever possible, lets the user make selections with CheckBoxes, ListBoxes, ComboBoxes, etc. instead of typing input.

The following is an example of a SAS program generated by EasySAS:

```
proc printto print='c:\temp\_SASFM.lst'
              log='c:\temp\_SASFM.log';
options linesize=72 pagesize=56 nodate
        pageno=1;
libname _DATAIN "d:\SAS\core\sample";
title 'Descriptive Statistics of salary';
proc means data= _DATAIN.salary
          Mean Sum STD maxdec=2;
var SALARY;
class JOBCODE;
run;
proc printto;
run;
```

Note that proc printto is used to redirect the log file and output file at the beginning of the program and also used to reset them to default at the end.

Development Procedure

For real business applications, it is recommended that developers use the following procedure:

- Develop and test SAS program(s) without any user interface or OLE Automation.
- Convert SAS program into SAS macro with well-defined input, output, and any additional parameters.
- Design a Windows user interface that matches the SAS macro.
- Dynamically generate calls to the SAS macro and then submit it to SAS through OLE automation.
- Test, test, and test again.

This approach separates the SAS program from user interface development thus make debugging much easier. This also lets a SAS analyst team up with a Windows developer to create an application without much cross training. It is also a big advantage to application maintenance since one can modify the SAS macro to improve the analysis with minimal modifications to the user interface.

Conclusion

This paper presented two approaches for using SAS as a database and analytic engine with a Windows user interface developed in Delphi, Visual Basic, or a similar tool. Both are portable and generic enough to be used in diverse applications. Companies already using SAS as an analytic tool can use these methods as a fast means to turn SAS programs into Windows applications. Companies doing traditional application development can use these same techniques to tap into powerful SAS analytic procedures. In either case developers can continue using familiar development tools.

Acknowledgments

Thanks to David Body of Big Creek Software for reviewing and editing this paper.

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. Windows 95, Windows NT, Visual C++, Visual Basic are registered trademarks of Microsoft Corporation. Delphi is a registered trademark of Borland International.

Author Contact

Zhuan (John) Xu

E-mail: johnxu@bigcreek.com
Phone: (515) 235-4403

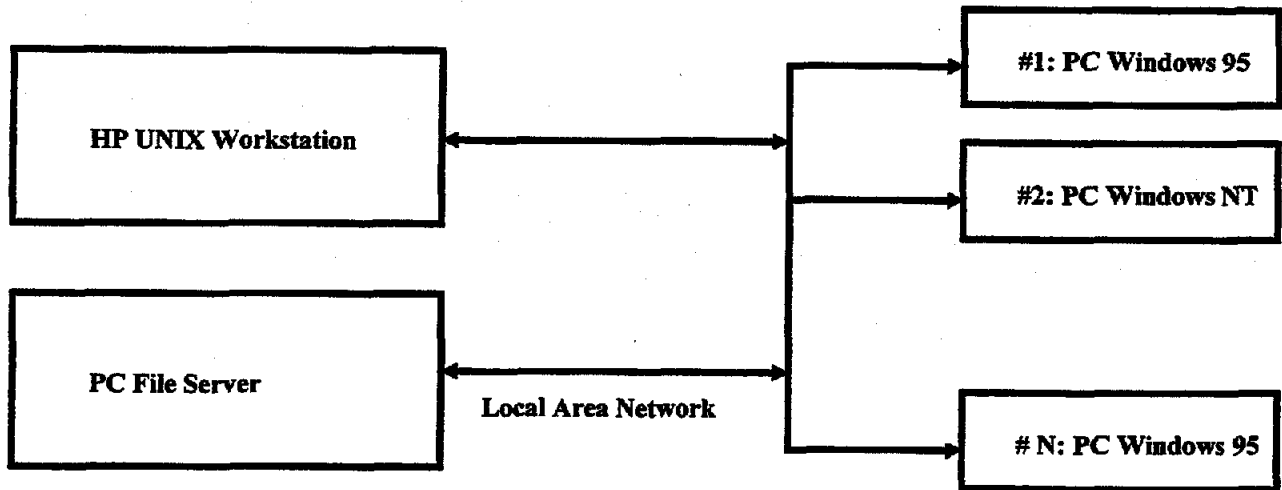


Figure 1, MAuditor Hardware

Database:	Analysis:	Communication:
Loading and cleaning raw data Create clustered index Accessory data Summary data Data extraction and conversion	Procedure Diagnosis Trending Provider Small Area Analysis	FTP Server UNIX Sever script Security

Figure 2, MAuditor Server Components

Medical Policy Database Search utilities Reporting/Graphing Mapping	Extracting data from server Setting up policy analysis Communication Components Dynamic query tool for detail/summary
--	--

Figure 3, MAuditor Front End Components

View Medical Policies

File Policy Analysis Help

Medical Policies: Policy Navigator

Policies	Effective	Revised	Description
19318	7/1/95	7/1/96	Reduction Mammoplasty* n covered code
36470	2/22/90	2/15/96	Sclerosing Injection of Varicose Veins**
43200	6/15/96	10/17/96	Upper Gastrointestinal Endoscopy* g

CPT Codes				Covered Diagnosis Codes			
Policy	First	Last	Group	Policy	First	Last	Group
43200	43200		1	43200	11284		1
43200	43202		1	43200	1500	1509	1
43200	43204		1	43200	1510	1519	1
43200	43205		1	43200	1520		1
43200	43215		1	43200	2110	2112	1
43200	43216		1	43200	2301	2302	1
43200	43217		1	43200	2307		1
43200	43219		1	43200	261		1
43200	43220		1	43200	2639		1
43200	43226		1	43200	2800	2809	1
43200	43227		1	43200	3071		1
43200	43228		1	43200	4560	45621	1
43200	43234		1	43200	5300	5309	1
43200	43235		1	43200	53100	53191	1

Status: Time:

Search

Description:

CPT	Description
10040	ACNE SURGERY
10060	DRAINAGE OF SKIN ABSCESS
10061	DRAINAGE OF SKIN ABSCESS
10080	DRAINAGE OF PILONIDAL CYST

Search Result:

Policy	First	Last	Group
10060	10060		1
10060	10061		1

Enter Code:

100

Type:

☒ CPT ☐ Diagnosis

☐ Description ☐ Description

Search Close

Figure 4, MAuditor Sample Screen 1-2

Set Up

Time Name: 95q12, 95q34, 96q12, 96q34

☐ Iowa Series Only
☒ BIDS of Iowa Only
☐ Exclude Lab
☐ Fetch Data Data

☒ CPU
☒ UPIN
☒ Diagnosis
☒ Small Area

☐ None
☐ By Year
☐ By Quarter
☐ By Month

OK Cancel

Part A Summary

EPT/HCP: From: To:
 Any Diagnosis: From: To:
 Provider: SUPIN:
 Prov ID:
 UPIN:
 Date:
 From: 00/00/0000
 To: 00/00/0000
 HIC:
 Small Area:
 Bene ZID:
 Revenue 1:
 Revenue 2:
 Revenue 3:
 Type of ID:
 Sort By:
 Order By: SArea, REV
☐ Desc
☐ Asc
 OK Cancel Print

SSAREA	REV	N	NUMBSRV	SUBMIT
0	0300	14350	14579	\$336,448.36
0	0301	1471	1478	\$36,491.83
0	0304	32	32	\$749.76
0	0305	1	1	\$23.50
0	0309	8	8	\$186.88
1	0300	197	197	\$4,622.64
1	0301	2	2	\$47.00
2	0300	575	577	\$13,455.42
2	0301	10	10	\$233.88
3	0300	236	236	\$5,548.24
3	0301	19	19	\$446.22
4	0300	139	139	\$3,266.91
5	0300	1855	1867	\$43,833.27
5	0301	2	2	\$47.56

Figure 5, MAuditor Sample Screen 3-4

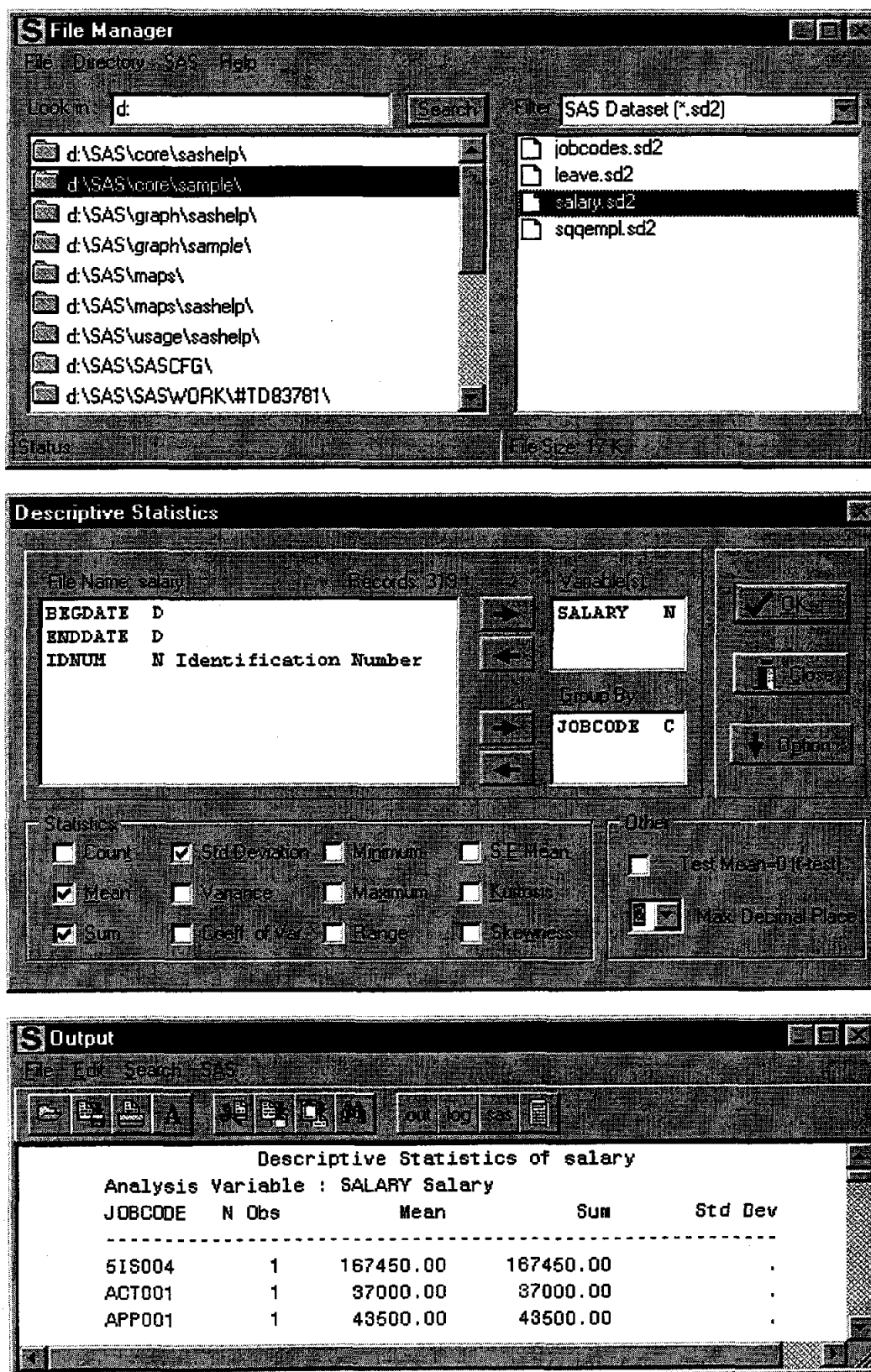


Figure 6, EasySAS Sample Screens 1-3