

Paper 266-30

SAS®9 Changes and Enhancements

Dana Rafiee, Destiny Corporation Rocky Hill, CT
Copyright© Destiny Corporation 2005

ABSTRACT

This document is an excerpt from Destiny Corporation's Version 9 Changes and Enhancement course materials, Copyright 2005. It is designed to give a brief overview of what Version 9 has to offer.

The primary goal of Version 9 is to provide support for a new level of computing that supports faster execution of applications, centralized access of data and support of the latest computing technology. These materials are designed as an overview of what is available and new and complement the online documentation that ships with the software.

INTRODUCTION

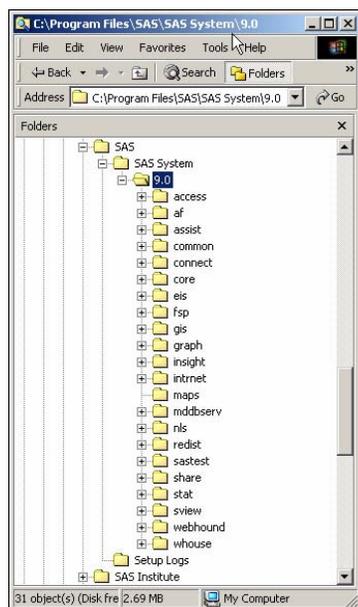
The new SAS Open Metadata Architecture is designed to allow all registered data in an organization to be centrally managed and accessed. This feature will ship with Version 9.1. The SAS Management Console will offer one point of control for all SAS servers and applications in the organization. The new multi-threaded architecture is automatically turned on to support most procedures that sort and summarize data. The Output Delivery System has been enhanced to support many new styles of markup, along with custom markup tag sets.

SAS also supports the industry standard Application Response Measurement (ARM) protocol for monitoring SAS processes. SAS V9 is designed to be upwardly compatible with code and data. Cross Environment Data Access (CEDA) is still supported on all operating systems. SAS Version 9 is designed to better support ADA 508 handicapped standards.

This course is designed to discuss as many issues as possible, from the installation of SAS to the finer aspects of operating under the new release.

INSTALLATION OF SAS

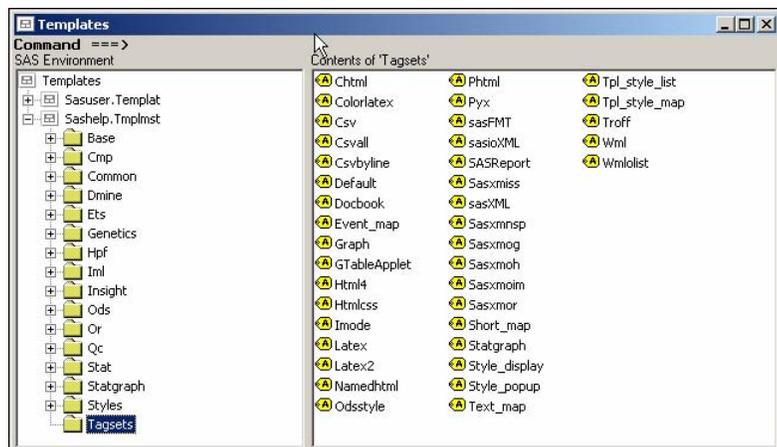
SAS is now installed in a new directory location. The following example shows the tree structure.



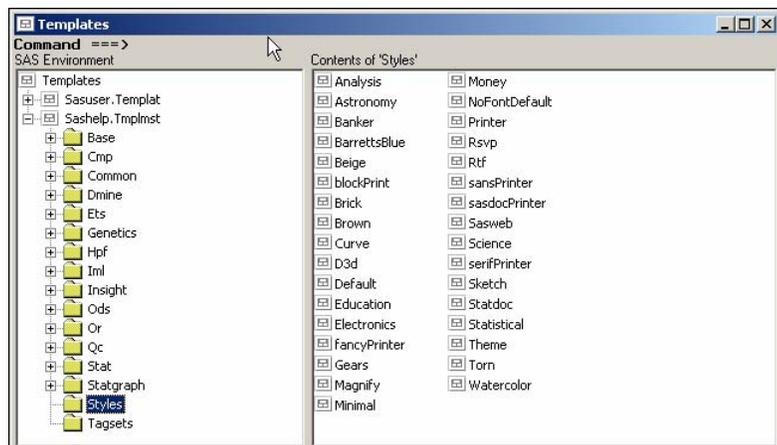
The new directory for Version 9 is SAS. SAS V9 is designed to live alongside SAS V8. Notice the SAS V9 directory and the SAS Institute V8 directory.

Note: SAS V9 and SAS V8 use some of the same shared components like the Enhanced Editor and ActiveX controls. When uninstalling one of the versions, do not uninstall the shared components.

software.



Select Styles to see all of the styles available.



ODS DOCUMENT VIEWER

The new ODS document viewer is now available. It can be used to see stored documents created with the ODS DOCUMENT statement. See the ODS section of this document for further details.



RETURN OF THE V6 WINDOWS

SAS brought back some of the old Version 6 windows in Display Manager per the request of users.

The commands to access them are:

- V6CAT
- V6LIB
- V6DIR
- V6VAR

DATA MIGRATION

In Version 9, SAS supports formats and informats that are longer than 8 bytes. The only difference in the data set structures is this capability. Version 8 data sets are upward compatible to Version 9 by default. Version 9 data sets are backward compatible if formats and informats conform to Version 8 naming conventions.

ENGINES

Libname statements offer support for the new V9 engine as well as the V8, V7 and V6 engines.

```
00001 libname version9 v9 'c:\';
00002 libname version8 v8 'c:\';
00003 libname version7 v7 'c:\';
00004 libname version6 v6 'c:\';
```

The file extensions are still the same as in Version 8.

File Extension	SAS Member Type	Description
.log	None	Log
.lst	None	Output
.sas	None	SAS Program
.sas7bacs	Access	Access descriptor
.sas7baud	Audit	Audit file
.sas7bcatalog	Catalog	SAS catalog
.sas7bdat	Table	Data set
.sas7bmdb	MDDDB	Multi-dimensional database
.sas7bndx	None	Data set index. Indexes are stored as separate files but are treated by the SAS System as integral parts of the SAS data file.
.sas7bods		Output Delivery System file
.sas7bpgm	Program	Stored DATA step program
.sas7bview	View	Data set view

NEW FORMATS

The new length for numeric format names is 32. For character names it is 31. Format names can be associated with a data set. The only difference between SAS V9 and previous versions of data sets is the existence of a format name longer than the traditional 8 bytes.

Version 9 data sets that use 8 byte or shorter format names can be read by Version 8.

\$BIDW.

This format is designed to convert a string to be logically or visually ordered. This works with Hebrew and Latin characters.

NEW INFORMATS

The new length for numeric informat names is 31 and for character names is 30. Informat names can be associated with a data set. The only difference between SAS V9 and previous versions of data sets is the existence of a format name longer than the traditional 8 bytes. Version 9 data sets that use 8 byte or shorter format names can be read by Version 8.

ANY DATE INFORMAT

Three new informats are now available to convert various date, time and datetime forms of data into a SAS date or SAS time. They are:

- ANYDTEw. To convert to a SAS date value
- ANYDTMEw. To convert to a SAS time value
- ANYDTMw. To convert to a SAS datetime value

These new informats were created to make reading these types of values simpler. It is important to realize that these informats make assumptions on a record by record basis. Ambiguous values can be interpreted in an incorrect fashion.

```

00001 data work.dates;
00002   infile cards;
00003   input @01 string $ 20.;
00004   extracteddate   = input(string,anydtde32.);
00005   extractedtime   = input(string,anydtme32.);
00006   extracteddatetime = input(string,anydtm32.);
00007   datalines;
00008 2002-Apr-15
00009 April 15, 2002
00010 April 15 2002
00011 2002Q2
00012 15APR2002
00013 2002110
00014 2002/04/15
00015 2002/04/15:11:22:33
00016 15APR2002:11:22:33
00017 12:34:56
00018 APR2002
00019 15/04/2002
00020 04/15/2002
00021 run;
00022 proc print data=work.dates;
00023 run;

```

Special Interpretations:

- Timezones with + or – GMT times will be ignored.
- The YEARCUTOFF= option interprets two digit years

DATESTYLE OPTION

The DATESTYLE option can be used when ambiguous values in dates exist. This option sets a default assumption for the date to be either DMY, MDY, or YMD.

```

00001 options datestyle=MDY;
00002
00003 data work.dates;
00004   infile cards;
00005   input @01 string $ 20.;
00006   extracteddate   = input(string,anydtde32.);
00007   extractedtime   = input(string,anydtme32.);
00008   extracteddatetime = input(string,anydtm32.);
00009   datalines;
00010 2002-Apr-15
00011 April 15, 2002
00012 April 15 2002
00013 2002Q2
00014 15APR2002
00015 2002110
00016 2002/04/15
00017 2002/04/15:11:22:33
00018 15APR2002:11:22:33
00019 12:34:56
00020 APR2002
00021 15/04/2002
00022 04/15/2002
00023 run;
00024 proc print data=work.dates;
00025 run;

```

NEW FUNCTIONS

There are several, new functions and call routines available in SAS. Most of these are designed for very specific manipulations of data.

PERL REGULAR EXPRESSIONS AND PATTERN MATCHING

There are several functions available to perform pattern matching routines on data. The following example shows a typical use for one of these functions. The PRXPARSE function is designed to specify pattern matching. This example sets YES/NO flags as to how phone numbers match a valid pattern.

```

00001 data work.prx;
00002   length parens dashes $3;
00003   if _N_ = 1 then do;
00004     retain parensx dashesx;
00005     parensx= prxparse("\/\([2-9]\d\d\)?[2-9]\d\d-\d\d\d\d\/");
00006     dashesx= prxparse("\/\d\d\d-\d\d\d-\d\d\d\d\/");
00007   end;
00008   length first last phone $ 16;
00009   input first last phone & $16.;
00010   if ^prxmatch(parensx, phone) then parens = 'NO';
00011   else parens = 'YES';
00012   if ^prxmatch(dashesx, phone) then dashes = 'NO';
00013   else dashes = 'YES';
00014   datalines;
00015   B.G. Johnson      (860)721-1684
00016   Sally Smith      888-456-1234
00017   Tommy Toad       (560) 454-6777
00018   Susan Infobeans  (765)142-3333
00019   Paulina Kent     (900)555-1212
00020 run;
00021 proc print data=work.prx;
00022 run;

```

Obs	parens	dashes	parensx	dashesx	first	last	phone
1	YES	NO	1	2	B.G.	Johnson	(860)721-1684
2	NO	YES	1	2	Sally	Smith	888-456-1234
3	NO	NO	1	2	Tommy	Toad	(560) 454-6777
4	NO	NO	1	2	Susan	Infobeans	(765)142-3333
5	YES	NO	1	2	Paulina	Kent	(900)555-1212

The valid list of values available for Perl expressions is available from WWW.PERLDOC.COM.

CALL SORTN/SORTC EXAMPLE

Call SORTN and CALL SORTC are quick ways to sort variable values inside the Data Step. It is not designed to replace PROC SORT. It is a simple way of ordering values of the same structure. For example, if it is used with character variables, they must be the same length.

Consider the following example. Four variables are loaded into a Data Step. We want them ordered smallest to largest and largest to smallest.

```

00001 data _null_;
00002   v1 = 16;
00003   v2 = 2;
00004   v3 = 4;
00005   v4 = 64;
00006   call sortn(of v1-v4);
00007   put _all_;
00008   call sortn(of v4-v1);
00009   put _all_;
00010 run;
00011 data _null_;
00012   v1 = '16';
00013   v2 = '2';
00014   v3 = '4';
00015   v4 = '64';
00016   call sortc(of v1-v4);
00017   put _all_;
00018   call sortc(of v4-v1);
00019   put _all_;
00020 run;

```

```

122 data _null_;
123   v1 = 16;
124   v2 = 2;
125   v3 = 4;
126   v4 = 64;
127   call sortn(of v1-v4);
128   put _all_;
129   call sortn(of v4-v1);
130   put _all_;
131 run;

NOTE: The SORTN function or routine is experimental in release 9.1.
v1=2 v2=4 v3=16 v4=64 _ERROR_=0 _N_=1
v1=64 v2=16 v3=4 v4=2 _ERROR_=0 _N_=1
NOTE: DATA statement used (Total process time):
   real time           0.00 seconds
   cpu time             0.01 seconds

132 data _null_;
133   v1 = '16';
134   v2 = '2';
135   v3 = '4';
136   v4 = '64';
137   call sortc(of v1-v4);
138   put _all_;
139   call sortc(of v4-v1);
140   put _all_;
141 run;

NOTE: The SORTC function or routine is experimental in release 9.1.
v1=16 v2=2 v3=4 v4=64 _ERROR_=0 _N_=1
v1=64 v2=4 v3=2 v4=16 _ERROR_=0 _N_=1
NOTE: DATA statement used (Total process time):
   real time           0.00 seconds
   cpu time             0.00 seconds

```

NEW DATA SET OPTIONS

Option	Definition
ENCODING=	This is designed for multinational language support, typically used with SAS/SHARE environments where data is shared between different countries. One data set can be referenced in one character set for one country and a different character set for a different country. See the table below for the encoding values supported.
ROLE=	When the data set is being used in a star schema style join, this table can be labeled FACT or DIMENSION. This can speed up processing if the appropriate tables are labeled. SAS will use the role identification during SQL joins.
SORTSEQ=	Determines the collating sequencing during sorting.
SPILL=NO/YES	This is an option on Data Step Views that tells SAS to produce or not produce spill files when a view is opened for two pass mode. This reduces the amount of disk space required for a spill file. See SAS Documentation for further information on this efficiency and when to use it.

ROLE= OPTION FOR OPTIMIZED JOINS

The following code is a simple example of how we can define particular tables as FACT or DIMENSION files.

```

00001 proc sql;
00002   select a.manager, a.title, a.dept, b.employee
00003   from saved.managers(role=fact) a,
00004        saved.employee(role=dim) b
00005   where a.dept=b.dept;
00006 quit;

```

Output - (Untitled)

Command ==>

MANAGER	TITLE	DEPT	EMPLOYEE
R OSWALD	ADMINISTRATOR	301	AL FRANKLIN
R OSWALD	ADMINISTRATOR	301	CRAIG MASTERS
R OSWALD	ADMINISTRATOR	301	FAT TUESDAY
J JONES	PRESIDENT	401	JIM DIXON
J JONES	PRESIDENT	401	JOHN DOE
J JONES	PRESIDENT	401	JOHN HOMES
S SMITH	VICE PRESIDENT	201	JP JONES
R OSWALD	ADMINISTRATOR	301	LARRY SMITH
J JONES	PRESIDENT	401	PAUL JONES
J JONES	PRESIDENT	401	PAUL IE SURE
S SMITH	VICE PRESIDENT	201	RICHARD NIXON
J JONES	PRESIDENT	401	SALLY MAY
R OSWALD	ADMINISTRATOR	301	STEVE SMITH

RESETTING THE SAS DATE ON OUTPUT

SAS introduces the DTRESET system option for resetting the date on SAS output.

```
00001 options date dtreset;
00002 ods rtf file='c:\sasdateupdate.rtf';
00003 proc print data=sashe lp.vmacro;
00004 run;
00005 ods rtf close;
```

06:04 Saturday, April 06, 2002

Obs	scope	name	offset	value
1	GLOBAL	SQL0BS	0	51
2	GLOBAL	SQL0OPS	0	1591

PUTLOG STATEMENT

The new PUTLOG statement is designed to always write information to the SAS log, no matter where the FILEREF points to. This can be handy for debugging. It is similar to the PUT statement, but does not reference the FILEREF destination.

OBJECT DOT SYNTAX

The Data Step is being extended. SAS has introduced the concept of Object Dot Syntax. This is similar to the concept of Dot Notation as applied to Version 8 of SAS Component Language (SCL).

DECLARE STATEMENT

The DECLARE statement has been added to the data step, along with this syntax. This allows declaration of an object. The subsequent syntax allows for methods to be called on that object: Object.method()

HASH TABLE FOR LOOKUPS

The first use of this new syntax has been introduced through Hash tables. Hash tables are a way of performing a table lookup by loading key variables and values into an array in memory and then matching those values to values being read from a data set.

The beauty of this feature is that the key information lives in memory and not on disk. This is another way of performing lookups that creates a similar result to using:

- Proc format
- Macro variables
- Arrays
- SQL Joins
- Indexing
- Data Step Merging

The hash table grows in memory based on the size of the data loaded. Consider the following example.

Managers Data Set

VIEWTABLE: saved.managers

	DEPT	MANAGER	TITLE
1	101	B WILLIE	MANAGER
2	201	S SMITH	VICE PRESIDENT
3	301	R OSWALD	ADMINISTRATOR
4	401	J JONES	PRESIDENT

Employee Data Set

	DEPT	EMPLOYEE
1	301	AL FRANKLIN
2	301	CRAIG MASTERS
3	301	FAT TUESDAY
4	401	JIM DIXON
5	401	JOHN DOE
6	401	JOHN HOWES
7	201	JP JONES
8	301	LARRY SMITH
9	401	PAUL JONES
10	401	PAULIE SURE
11	201	RICHARD NIXON
12	401	SALLY MAY
13	301	STEVE SMITH
14	501	ELIZABETH DOLE

The following syntax will read both files.

```

00001 data work.hash;
00002   length manager $20 title $50;
00003   if _n_ = 1 then do;
00004     declare associativearray aa(dataset: "saved.managers");
00005     aa.defineKey('DEPT');
00006     aa.defineData('MANAGER', 'TITLE');
00007     aa.defineDone();
00008   end;
00009   set saved.employee;
00010   if aa.find() = 0;
00011 run;
00012 proc print data=work.hash;
00013 run;

```

And yield the following result.

Obs	manager	title	DEPT	EMPLOYEE
1	R OSWALD	ADMINISTRATOR	301	AL FRANKLIN
2	R OSWALD	ADMINISTRATOR	301	CRAIG MASTERS
3	R OSWALD	ADMINISTRATOR	301	FAT TUESDAY
4	J JONES	PRESIDENT	401	JIM DIXON
5	J JONES	PRESIDENT	401	JOHN DOE
6	J JONES	PRESIDENT	401	JOHN HOWES
7	S SMITH	VICE PRESIDENT	201	JP JONES
8	R OSWALD	ADMINISTRATOR	301	LARRY SMITH
9	J JONES	PRESIDENT	401	PAUL JONES
10	J JONES	PRESIDENT	401	PAULIE SURE
11	S SMITH	VICE PRESIDENT	201	RICHARD NIXON
12	J JONES	PRESIDENT	401	SALLY MAY
13	R OSWALD	ADMINISTRATOR	301	STEVE SMITH

MULTI-THREADED ARCHITECTURE

One of the biggest enhancements with SAS software in Version 9 is its ability to support multi-threaded access to files for use in the Data Step and certain procedures. The concept is simple. Instead of using the traditional 'serial' approach to either sorting or summarizing data, SAS now breaks up the data into smaller chunks, performs the operation and then puts the result back together.

SORTING ANALOGY

Consider the following analogy for sorting. There are four decks of playing cards. The goal is to order them. One person can try to order all four decks together to produce an ordered result. Another way of doing this is to get four people, each tasked with ordering one deck. The ordered four decks are then combined for a final result.

SUMMARIZING ANALOGY

The goal is to get a total of 100 numbers. One person can sit down with a pencil and paper and total up the 100 numbers to produce a result. Another way of doing this is to get four people to each take 25 numbers and produce 4 totals. The 4 totals are then added up to produce the number.

The division of labor/tasks in these examples demonstrates why a multi-threaded architecture makes sense when possible.

Multi-threading is supported for:

- PROC SORT
- PROC SUMMARY
- PROC MEANS
- PROC REPORT
- PROC TABULATE
- PROC SQL
- PROC REG
- PROC GLM
- PROC ROBUSTREG
- PROC LOESS
- PROC DMREG
- PROC DMINE
- PROC SERVER

By default, multi-threading is turned on in Version 9 for all of these procedures. Therefore, there is a new option available with each procedure (THREADS/NOTHREADS) to optionally turn this feature off.

CPUCOUNT OPTION

It is important to realize that multi-threading works best in a multiple processor environment. For example, if SAS is running on a four processor server, it will attempt to utilize all of the CPUs available. For large SAS jobs, this may impact performance of other applications or programs running on that server.

The default value of CPUCOUNT is set to the maximum number of CPUs found. This tells SAS to go and use as many processors as it can access. In some situations, it may be wise to limit the number of CPUs accessed by SAS by setting this value to a maximum CPU number. CPUCOUNT is a way to throttle back the number of processors used in multiple processor environments.

BENCHMARKING

In threading environments, it is possible that the CPU time used to process may actually be larger than real time/wall time. This may be a consideration when scheduling large production jobs.

SCALABLE PERFORMANCE DATA ARCHITECTURE

Since Version 6 of SAS, the Scalable Performance Data Server has been available as a separate product that allows for breaking apart data storage to speed up I/O and avoid the traditional 'serial based' I/O architecture. SPDS Software also supported a very intense security model for access to this data. The ideal storage of a data source would be split across several storage locations that were usually separate disks with separate disk controllers.

In Version 9, this entire architecture, except for the security model, is included.

SCALABLE PERFORMANCE DATA ENGINE SPDE

This is a new engine that is part of Base SAS software. It is designed to be used on a libname statement for much quicker access to data stored on disk. The ideal environment is the same as listed above.

- Each data location is stored on a separate disk.
- Each disk has a separate disk controller.
- Each metadata repository is stored in a separate location.
- Each index is stored in a separate location.

SCALABLE ANALOGY

Consider a two lane highway with two toll booths. Traffic can proceed through those toll booths at a particular speed.

Now consider a two lane highway with 10 toll booths. People split apart to pay the tolls and then merge back together into a two lane highway.

Now consider a 10 lane highway with 10 toll booths.

Now consider a 10 lane highway with no toll booths.

Which one would be fastest?

SPDE performs best when everything is separated into its specific tasks with no stopgaps to slow down processes e.g. a shared disk controller (bad). Performance is best on a 10 lane highway with no toll booths.

Consider the following example to demonstrate the syntax.

```
00001 libname spdeloc spde 'c:\v9\metadata'
00002         indexpath=('c:\v9\index')
00003         datapath=('c:\v9\loc1', 'c:\v9\loc2', 'c:\v9\loc3');
00004
00005 options validvarname=any partsize=2000000;
00006
00007 data spdeloc.newindex(index=(i));
00008     do i = 1 to 1000000;
00009         'Some Big Variable'n = 'This is my value';
00010         output;
00011     end;
00012 run;
00013 proc contents data=spdeloc.newindex;
00014 run;
```

This form of the libname statement uses the SPDE engine.

There is a separate location for:

- Metadata
- Datapath1
- Datapath2
- Datapath3
- Indexes

This example is for syntax only. In an ideal situation, the locations would all reside on separate disks with different disk controllers.

CONCLUSION

Large amounts of data can be processed effectively with this architecture. The location of data is now split and must be maintained. This is a consideration when moving data. However, the performance gained may be worth the additional housekeeping.

NEW/ENHANCED PROCEDURES

PROC SQL

PROC SQL now has a THREADS/NOTTHREADS option to turn multi-threading on and off.

A SAS data set can be reference by the real, physical location.

```
00001 proc sql;
00002     select a.manager, a.title, a.dept, b.employee
00003     from 'c:\v9\managers.sas7bdat'(role=fact) a,
00004         'c:\v9\employee.sas7bdat'(role=dim) b
00005     where a.dept=b.dept;
00006 quit;
```

Leading zeros are supported when using the INTO clause.

```
00001 proc sql noprint;
00002     select distinct dept into :d01 - :d04
00003     from saved.managers;
00004 quit;
00005 proc print data=sashelp.vmacro;
00006     where name in ('D01', 'D02', 'D03', 'D04');
00007 run;
```

MACRO ENHANCEMENTS

CALL SYMPUTX MACRO STATEMENT

This is a new statement that creates a macro variable at execution time in the data step by:

- Left justifying the value
- Trimming trailing blanks
- Automatically converting numeric values to character

```

Log - (Untitled)
Command ==> |
1  data _null_;
2  number = 99;
3  * Traditional lazy/messy way;
4  call symput('m1',number);
5  * You realize you need to do an explicit conversion;
6  call symput('m2',put(number,8.));
7  * You also want to left justify it;
8  call symput('m3',left(put(number,8.)));
9  * You also want to trim it;
10 call symput('m4',trim(left(put(number,8.))));
11 * The version 9 way;
12 call symputx('m5',number);
13 run;

NOTE: Numeric values have been converted to character values at the places given by:
      (Line):(Column).
      4:21
NOTE: DATA statement used (Total process time):
      real time    0.79 seconds
      cpu time     0.23 seconds

14 data _null_;
15 put 'x&m1.x';
16 put 'x&m2.x';
17 put 'x&m3.x';
18 put 'x&m4.x';
19 put 'x&m5.x';
20 run;

x          99x
x      99x
x99      x
x99x
x99x
NOTE: DATA statement used (Total process time):
      real time    0.13 seconds
      cpu time     0.12 seconds

```

NEW ODS STATEMENTS

CHTML STATEMENT

This statement creates the simplest HTML possible without using styles.

CSV STATEMENT

This statement is designed to create a comma delimited CSV file of table information. These types of files are typically imported into Excel.

CSVALL STATEMENT

The statement is designed to create a CSV file while preserving titles, notes and bylines.

FORMATTED EXCEL TIP

Output from SAS can create formatted data that Excel can read. Consider creating an HTML file with ODS, an XLS extension and then opening it up in Excel.

DOCBOOK STATEMENT

This statement creates XML files and supports the DocBook DTD format from Oasis.

DOCUMENT STATEMENT

This statement is designed to change the order or type of display of any output through ODS without having to rerun the procedures.

HTMLCSS STATEMENT

This statement is designed to create a Cascading Style Sheet document from an existing SAS style sheet, alongside HTML. It will also use an existing Cascading Style sheet, if specified.

IMODE STATEMENT

This statement produces HTML in a column form that is separated by lines.

LATEX STATEMENT

This statement produces LaTeX output for high quality typesetting systems.

MARKUP STATEMENT

This is an example that allows for custom tag set creation. Any customized form of tag sets can be created, registered and used in ODS.

PCL STATEMENT

This statement is designed to create information for HP LaserJet emulation.

TROFF STATEMENT

This statement creates Troff markup language for high quality laser printing and typesetting.

WML STATEMENT

This statement is designed to create Wireless Markup Language for WAP based (phone display) environments with an HREF table of contents.

WMLLIST STATEMENT

This statement is designed to also create Wireless Markup Language with a table of contents option list.

NEW ODS OPTIONS

There are several new options in ODS. They are typically designed for better control and formatting of output.

COLUMNS OPTION

This option is designed to create multiple columns in output.

```
00001 title 'COLUMNS= Example';
00002 ods pdf file='c:\columns.pdf' startpage=no columns=2;
00003 ods rtf file='c:\columns.rtf' startpage=no columns=2;
00004 proc print data=sashelp.vmacro;
00005 run;
00006 ods _all_ close;
```

PDF Output

COLUMNS= Example

Obs	scope	name	Obs	scope	name
1	GLOBAL	M1	19	AUTOMATIC	SYSDEVIC
2	GLOBAL	M2	20	AUTOMATIC	SYSDMG
3	GLOBAL	M3	21	AUTOMATIC	SYSDSN
4	GLOBAL	M4	22	AUTOMATIC	SYSENV
5	GLOBAL	M5	23	AUTOMATIC	SYSERR
6	AUTOMATIC	AFDSID	24	AUTOMATIC	SYSFILRC
7	AUTOMATIC	AFDSNAME	25	AUTOMATIC	SYSINDEX
8	AUTOMATIC	AFLIB	26	AUTOMATIC	SYSINFO
9	AUTOMATIC	AFSTR1	27	AUTOMATIC	SYSJOBID
10	AUTOMATIC	AFSTR2	28	AUTOMATIC	SYSLAST
11	AUTOMATIC	FSPBDV	29	AUTOMATIC	SYSLCKRC
12	AUTOMATIC	SYSBUFFR	30	AUTOMATIC	SYSLIBRC
13	AUTOMATIC	SYSCC	31	AUTOMATIC	SYSMACRONAME
14	AUTOMATIC	SYSCHARWIDTH	32	AUTOMATIC	SYSMAXLONG
15	AUTOMATIC	SYSCMD	33	AUTOMATIC	SYSMENV
16	AUTOMATIC	SYSDATE	34	AUTOMATIC	SYSMSG
17	AUTOMATIC	SYSDATE9	35	AUTOMATIC	SYSNCPU
18	AUTOMATIC	SYSDAY	36	AUTOMATIC	SYS Parm

RTF Output

COLUMNS= Example

Obs	scope	name	Obs	scope	name
1	GLOBAL	M1	36	AUTOMATIC	SYS Parm
2	GLOBAL	M2	37	AUTOMATIC	SYSBUFFR
3	GLOBAL	M3	38	AUTOMATIC	SYSPROCESSID
4	GLOBAL	M4	39	AUTOMATIC	SYSPROCESSNAME
5	GLOBAL	M5	40	AUTOMATIC	SYSPROCNAME
6	AUTOMATIC	AFDSID	41	AUTOMATIC	SYSRC

MARGIN AND INDENT OPTIONS

This option is designed to control margins and indentations in output.

```
00001 options orientation=landscape;
00002 ods pdf file='c:\marginindent.pdf' startpage=no columns=1;
00003 ods rtf file='c:\marginindent.rtf' startpage=no columns=1;
00004 proc report data=sashelp.bpl nowd;
00005   column patient comment1 comment2 comment3 comment4;
00006   define patient / group;
00007   define comment1 / computed style={cellwidth=200pt} 'Normal width';
00008
00009   define comment2 / computed
00010     style={leftmargin=12pt} 'LeftMargin justified';
00011   define comment3 / computed
00012     style={leftmargin=12pt indent=12pt}
00013     'LeftMargin indented';
00014   define comment4 / computed
00015     style={leftmargin=12pt indent=-12pt}
00016     'LeftMargin reverse(negative) indent';
00017
00018   compute comment1 / length=50 character;
00019     comment1 = "The patient's adverse events were moderate. Outcome=";
00020   endcomp;
00021   compute comment2 / length=50 character;
00022     comment2 = "The patient's adverse events were mild. Outcome=cure";
00023   endcomp;
00024   compute comment3 / length=50 character;
00025     comment3 = "The patient had no adverse events. Outcome=NA.";
00026   endcomp;
00027   compute comment4 / length=50 character;
00028     comment4 = "The patient changed dosage. Removed from the study.";
00029   endcomp;
00030 run;
00031 ods _all_ close;
```

PDF Output

COLUMNS= Example

Patient Number	Normal width	LeftMargin justified	LeftMargin indented	LeftMargin reverse(negative) indent
203	The patient's adverse events were moderate. Outcome	The patient's adverse events were mild. Outcome=CU	The patient had no adverse events. Outcome=NA	The patient changed dosage. Removed from the study
204	The patient's adverse events were moderate. Outcome	The patient's adverse events were mild. Outcome=CU	The patient had no adverse events. Outcome=NA	The patient changed dosage. Removed from the study
206	The patient's adverse events were moderate. Outcome	The patient's adverse events were mild. Outcome=CU	The patient had no adverse events.	The patient changed dosage. Removed from the study

RTF Output

COLUMNS= Example

Patient Number	Normal width	LeftMargin justified	LeftMargin indented	LeftMargin reverse(negative) indent
203	The patient's adverse events were moderate. Outcome	The patient's adverse events were mild. Outcome=CU	The patient had no adverse events. Outcome=NA	The patient changed dosage. Removed from the study
204	The patient's adverse events were moderate. Outcome	The patient's adverse events were mild. Outcome=CU	The patient had no adverse events. Outcome=NA	The patient changed dosage. Removed from the study
206	The patient's adverse events were moderate. Outcome	The patient's adverse events were mild. Outcome=CU	The patient had no adverse events.	The patient changed dosage. Removed from the study

TEXT OPTION

This option is designed to allow placement of text in any location around ODS output.

```
00001 title Text Demonstration;
00002 ods pdf file='c:\v9\text.pdf' startpage=no text='Text before the report that might describe the upcoming information';
00003 ods rtf file='c:\v9\text.rtf' startpage=no text='Text before the report that might describe the upcoming information';
00004 proc print data=sashelp.vmacro;
00005 run;
00006 ods pdf text='Text after the report. Remember, this is not a footnote.';
00007 ods rtf text='Text after the report. Remember, this is not a footnote.';
00008 ods _all_ close;
```

PDF Output

Text Demonstration

Text before the report that might describe the upcoming information

50	AUTOMATIC	SYSVLONG	0	9.00.00BOP031302
51	AUTOMATIC	SYSVLONG4	0	9.00.00BOP03132002

Text after the report. Remember, this is not a footnote.

ORIENTATION OPTION

This option allows the changing of output orientation. Notice the placement of the system option.

```
00001 options orientation = landscape;
00002 ods rtf file = 'c:\v9\portrait.rtf';
00003 proc print data = sashelp.vmacro;
00004 run;
00005 options orientation = portrait;
00006 * Notify ODS RTF that the orientation is now portrait;
00007 ods rtf;
00008 proc print data = sashelp.vmacro;
00009 run;
00010 ods rtf close;
```

Obs	Country	Income	Other	Value
1	USA	10000	10000	10000
2	USA	20000	20000	20000
3	USA	30000	30000	30000
4	USA	40000	40000	40000
5	USA	50000	50000	50000
6	USA	60000	60000	60000
7	USA	70000	70000	70000
8	USA	80000	80000	80000
9	USA	90000	90000	90000
10	USA	100000	100000	100000
11	USA	110000	110000	110000
12	USA	120000	120000	120000
13	USA	130000	130000	130000
14	USA	140000	140000	140000
15	USA	150000	150000	150000
16	USA	160000	160000	160000
17	USA	170000	170000	170000
18	USA	180000	180000	180000
19	USA	190000	190000	190000
20	USA	200000	200000	200000
21	USA	210000	210000	210000
22	USA	220000	220000	220000
23	USA	230000	230000	230000
24	USA	240000	240000	240000
25	USA	250000	250000	250000
26	USA	260000	260000	260000
27	USA	270000	270000	270000
28	USA	280000	280000	280000
29	USA	290000	290000	290000
30	USA	300000	300000	300000
31	USA	310000	310000	310000
32	USA	320000	320000	320000
33	USA	330000	330000	330000
34	USA	340000	340000	340000
35	USA	350000	350000	350000
36	USA	360000	360000	360000
37	USA	370000	370000	370000
38	USA	380000	380000	380000
39	USA	390000	390000	390000
40	USA	400000	400000	400000
41	USA	410000	410000	410000
42	USA	420000	420000	420000
43	USA	430000	430000	430000
44	USA	440000	440000	440000
45	USA	450000	450000	450000
46	USA	460000	460000	460000
47	USA	470000	470000	470000
48	USA	480000	480000	480000
49	USA	490000	490000	490000
50	USA	500000	500000	500000
51	USA	510000	510000	510000
52	USA	520000	520000	520000
53	USA	530000	530000	530000
54	USA	540000	540000	540000
55	USA	550000	550000	550000
56	USA	560000	560000	560000
57	USA	570000	570000	570000
58	USA	580000	580000	580000
59	USA	590000	590000	590000
60	USA	600000	600000	600000
61	USA	610000	610000	610000
62	USA	620000	620000	620000
63	USA	630000	630000	630000
64	USA	640000	640000	640000
65	USA	650000	650000	650000
66	USA	660000	660000	660000
67	USA	670000	670000	670000
68	USA	680000	680000	680000
69	USA	690000	690000	690000
70	USA	700000	700000	700000
71	USA	710000	710000	710000
72	USA	720000	720000	720000
73	USA	730000	730000	730000
74	USA	740000	740000	740000
75	USA	750000	750000	750000
76	USA	760000	760000	760000
77	USA	770000	770000	770000
78	USA	780000	780000	780000
79	USA	790000	790000	790000
80	USA	800000	800000	800000
81	USA	810000	810000	810000
82	USA	820000	820000	820000
83	USA	830000	830000	830000
84	USA	840000	840000	840000
85	USA	850000	850000	850000
86	USA	860000	860000	860000
87	USA	870000	870000	870000
88	USA	880000	880000	880000
89	USA	890000	890000	890000
90	USA	900000	900000	900000
91	USA	910000	910000	910000
92	USA	920000	920000	920000
93	USA	930000	930000	930000
94	USA	940000	940000	940000
95	USA	950000	950000	950000
96	USA	960000	960000	960000
97	USA	970000	970000	970000
98	USA	980000	980000	980000
99	USA	990000	990000	990000
100	USA	1000000	1000000	1000000

PAGE X OF Y SUPPORT

SAS now supports the ability to put page numbers on output with the total pages. This is currently supported for RTF.

```
00001 ods escapechar = '\';
00002 title 'Page X of Y example.' j=r 'Page \{pageof}';
00003 title2 This works in RTF only.;
00004 ods rtf file='c:\pagexofy.rtf';
00005 proc print data=sasHELP.VMACRO;
00006 run;
00007 ods rtf close;
```

Obs	scope	name	offset	value
1	GLOBAL	M1	0.99	
2	GLOBAL	M2	0.99	
3	GLOBAL	M3	0.99	
4	GLOBAL	M4	0.99	

NEW ODS STYLES

DECIMAL ALIGNMENT

```
00001 data work.decimal;
00002 input @01 charvar $15.
00003 @16 numvar 6.;
00004 cards;
00005 123 ( 4.8%) 42.6
00006 45 ( 4.9%) 456.99
00007 6 ( 5.7%) .
00008 789 ( 1.6%) 88
00009 10 ( 11.4%) 5.0
00010 run;
00011 ods rtf file='decimal1.rtf';
00012 title Regular Listing;
00013 proc print data=work.decimal;
00014 run;
00015 title Decimal Aligned Listing;
00016 proc print data=work.decimal;
00017 var charvar numvar / style(COLUMN)={just=d}
00018 ;
00019 run;
00020 ods rtf close;
```

Obs	charvar	numvar
1	123 (4.8%)	42.60
2	45 (4.9%)	456.99
3	6 (5.7%)	.
4	789 (1.6%)	88.00
5	10 (11.4%)	5.00

Obs	charvar	numvar
1	123 (4.8%)	42.60
2	45 (4.9%)	456.99
3	6 (5.7%)	.
4	789 (1.6%)	88.00
5	10 (11.4%)	5.00

MP CONNECT

This enhances traditional SAS/Connect software with asynchronous processing that is now production. Asynchronous processing is available on remotely submitted servers or processes on multiple processor systems. MP CONNECT processing can increase the performance of processes.

TRADITIONAL EXAMPLE

This is a simple example of how MP CONNECT can work.

```

00001 /* Global SAS system options to star an MP CONNECT remote session. */
00002 option autosignon=yes;
00003 option sascmd="sas";
00004
00005 /* Remote submit first task. */
00006 rsubmit process=task1 wait=no;
00007
00008     libname temp 'c:\v9';
00009     data work.class;
00010         do i = 1 to 1000000;
00011             name = 'SAS';
00012             output;
00013         end;
00014     run;
00015     proc sort data=work.class out=temp.class1;
00016         by descending i;
00017     run;
00018
00019 endrsubmit;
00020
00021 /* Remote submit second task. */
00022 rsubmit process=task2 wait=no;
00023
00024     libname temp 'c:\v9';
00025     data work.class;
00026         do i = 1 to 1000000;
00027             name = 'SAS';
00028             output;
00029         end;
00030     run;
00031     proc sort data=work.class out=temp.class2;
00032         by descending i;
00033     run;
00034
00035 endrsubmit;
00036
00037 /* Wait for both tasks to complete. */
00038 waitfor _all_ task1 task2;

```

```

00040 /* Get the remote logs and place them in the current log */
00041 rget task1;
00042 rget task2;
00043
00044 /* Merge the results and continue processing. */
00045 libname temp 'c:\v9';
00046 data work.sorted;
00047 merge temp.class1 temp.class2;
00048 run;

```

```

Log (Untitled)
Command ==>
86 /* Global SAS system options to star an MP CONNECT remote session. */
87 option autosignon=yes;
88 option sascmd="sas";
89
90 /* Remote submit first task. */
91 rsubmit process=task1 wait=no;
NOTE: Remote signon to TASK1 commencing (SAS Release 9.00.00B0P031302).
NOTE: Copyright (c) 2002 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software Pre-Production Version 9.00 (TS B0)
      Licensed to SAS Institute, Internal Test Release, Site 0000000001.
NOTE: This session is executing on the WIN_PRO platform.

NOTE: SAS initialization used:
      real time      4.24 seconds
      cpu time       0.62 seconds

NOTE: Remote signon to TASK1 complete.
NOTE: Background remote submit to TASK1 in progress.
92
93 /* Remote submit second task. */
94 rsubmit process=task2 wait=no;
NOTE: Remote signon to TASK2 commencing (SAS Release 9.00.00B0P031302).
NOTE: Copyright (c) 2002 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software Pre-Production Version 9.00 (TS B0)
      Licensed to SAS Institute, Internal Test Release, Site 0000000001.
NOTE: This session is executing on the WIN_PRO platform.

NOTE: SAS initialization used:
      real time      1.94 seconds
      cpu time       0.56 seconds

NOTE: Remote signon to TASK2 complete.
NOTE: Background remote submit to TASK2 in progress.
95
96 /* Wait for both tasks to complete. */
97 waitfor _all_ task1 task2;
98
99 /* Get the remote logs and place them in the current log */
100 rget task1;
NOTE: Remote submit to TASK1 commencing.

```

Notice the RGET statements to allow the local SAS log to see the remote SAS log information.

Notice the WAITFOR statements which must be used at strategic points in processing.

PIPING EXAMPLE

Piping is a new methodology in SAS where the output of one step is automatically being fed to the input of a subsequent step. When the process is appropriate, for example, the creation of a data set with a data step feeding directly into a subsequent Proc Sort, performance increases are possible. The output of the first process is not written to disk. The input of the second process does not read from disk. All information is passed through memory.

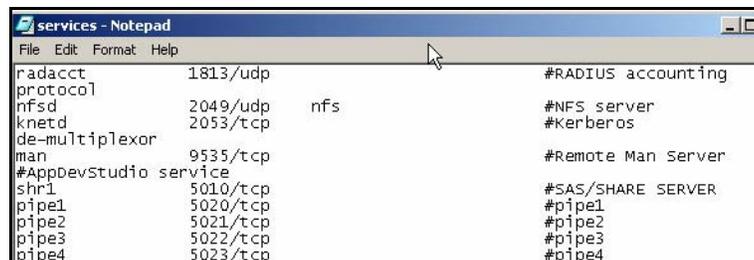
This is a simple example of how piping might be used. Notice the use of real physical storage to pass data between sessions.

```

00001 /* Global SAS system options to star an MP CONNECT remote session. */
00002 libname temp 'c:\v9';
00003 option autosignon=yes;
00004 option sascmd="sas";
00005 signon task1 sascmd="!sascmd";
00006 signon task2 sascmd="!sascmd";
00007 /* Remote submit first task. */
00008 rsubmit process=task1 wait=no;
00009     libname pipe1 sassock ":pipe1";
00010     data pipe1.class1;
00011         do i = 1 to 1000000;
00012             name = 'SAS';
00013             output;
00014         end;
00015     run;
00016 endrsubmit;
00017
00018 libname pipe1 sassock ":pipe1";
00019 proc sort data=pipe1.class1 out=temp.class1;
00020     by descending i;
00021 run;
00022
00023 rsubmit process=task2 wait=no;
00024     libname pipe2 sassock ":pipe2";
00025     data pipe2.class2;
00026         do i = 1 to 1000000;
00027             name = 'SAS';
00028             output;
00029         end;
00030     run;
00031 endrsubmit;
00032
00033 libname pipe2 sassock ":pipe2";
00034 proc sort data=pipe2.class2 out=temp.class2;
00035     by descending i;
00036 run;
00037
00038 /* Wait for both tasks to complete. */
00039 waitfor _all_ task1 task2;
00040
00041 /* Get the remote logs and place them in the current log */
00042 rget task1;
00043 rget task2;
00044
00045 /* Merge the results and continue processing. */
00046 data work.sorted;
00047     merge temp.class1 temp.class2;
00048 run;

```

In addition, the pipe locations must be listed in the services file. Notice pipe1 – pipe4 in the file listed below.



Service Name	Port	Protocol	Additional Info	Comment
radacct	1813	udp		#RADIUS accounting
protocol				
nfsd	2049	udp	nfs	#NFS server
knetd	2053	tcp		#Kerberos
de-multiplexor				
man	9535	tcp		#Remote Man Server
#AppDevstudio service				
shr1	5010	tcp		#SAS/SHARE SERVER
pipe1	5020	tcp		#pipe1
pipe2	5021	tcp		#pipe2
pipe3	5022	tcp		#pipe3
pipe4	5023	tcp		#pipe4

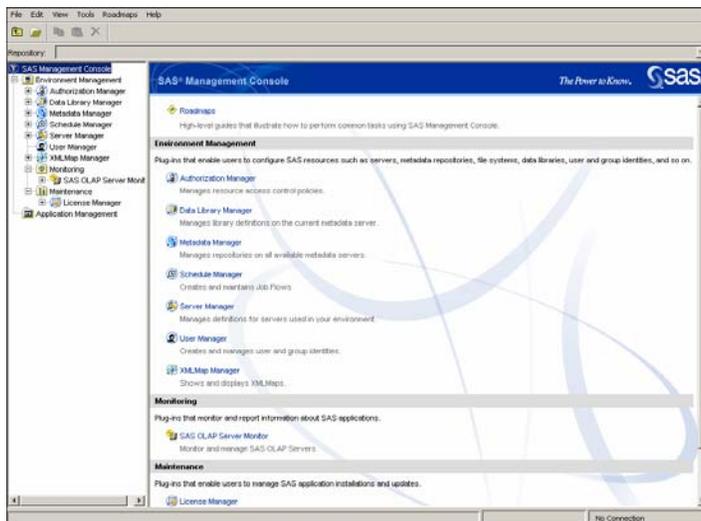
SAS MANAGEMENT CONSOLE

The SAS Management Console is designed to allow for managing and monitoring metadata, servers, libraries of data, servers and security from one, central point in an organization.

This application also supports third part plug-ins.

ASPECTS OF THE MANAGEMENT CONSOLE INCLUDE:

- Server Manager – manage SAS servers
- Metadata Manager –interact with running repository servers
- Application Manager – SAS application plug-ins
- SAS Library Manager – define and manage SAS libraries
- User Manager – define SAS Users and Groups
- Authorization Manager - administer authorization policy for accessing SAS Metadata and OLAP



THE BENEFITS OF THE SAS MANAGEMENT CONSOLE ARE:

- Simplify administrative tasks by using the same tools for all SAS products and solutions.
- Reduce staff training and support time.
- Build standard and repeatable processes for SAS operations.
- Define and manage connections to servers:
 - Application
 - Database
 - SAS
 - IOM Bridge
 - Others
- Define and manage SAS users and groups.
- Policy descriptions: set values for user and group attributes (such as read, write, etc.)
- Manage SAS Library definitions.
- Manage Database Schemas.
- Administer authorization policy for accessing SAS Metadata and OLAP:
 - Permission Creation
 - Access Control Templates
 - Resource Authorization Definitions
- Supports plug-ins to administer SAS applications, or user-written SAS applications.
- Manage SAS licenses (SETINIT):
 - The SAS License Manager (SLM) displays SAS installed software information.
- Monitor SAS processes:
 - Determine where SAS is running
 - Identify user of a SAS process
- Identify “orphans”
- Interface with resource managers

OPERATING SYSTEMS SUPPORTED

SAS V9 is supported under all of the Windows operating systems, including NT, 2000 and XP, but not supported under Windows 95, 98 and Me.

Operating System	Size
Windows NT4.0/2000/XP (WNT/W2K/WXP)	32 bit
OpenVMS Alpha 7.2	64 bit
Compaq's Digital UNIX 5.1	64 bit
HP HP-UX 64bit 11.0	64 bit
Solaris 64bit Solaris8	64 bit
AIX 64bit 5.1	64 bit
RedHat Linux 7.2 on Intel	32 bit
OS/390 (MVS) V2R10	32 bit

CONCLUSION

As you can see, SAS Version 9 offers a whole new way of processing that yields more choices to take your programming to the next level.

TRADEMARK CITATION

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

COPYRIGHT INFORMATION:

Copyright 2005 Destiny Corporation. All rights reserved.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please contact:

Dana Rafiee
Destiny Corporation
2075 Silas Deane Highway, Rocky Hill, CT 06067-2338
Phone: 800-700-TRAINING; (860) 721-1684
Fax: (860) 721-9784
Email: info@destinycorp.com
Web: www.destinycorp.com