TS-566A

Processing DB2/6000® Dates with SAS/ACCESS® in the UNIX® Environment

Technical Support Division
UNIX Systems Interface

SAS Institute, Inc.
SAS Campus Drive
Cary, N.C. 27513

Abstract

This paper is meant to document and clarify the issues involved when dealing with date conversion for DB2/6000 on UNIX platforms. Due to the increased interest of the SAS/ACCESS Interface to DB2/6000 product in the UNIX environment, this has been a popular topic for a number of SAS users.

Introduction

*The ACCESS Procedure is not available for the DB2 interface. The SAS/ACCESS Interface to DB2 product *only* provides support for the SQL Procedure Pass-Through Facility and the DBLOAD Procedure. With SAS Release 7.x, the LIBNAME engine support will be added to the SAS/ACCESS product and the date conversion issues discussed here will also follow the same rules.*

There are varying data types for different Database Management Systems (DBMSs) for which the SAS/ACCESS product will perform default format conversions.

Every DBMS column in a table has a name and a data type. The data type indicates to the DBMS how much physical storage to reserve for the column and the form in which the data are stored.

The DATE data type is handled in a different manner across different DBMSs. The DB2/6000 examples that follow were run on an AIX/RS6000 system, using SAS 6.12 TS020 and SAS Version 7. These demonstrate the use of dates using the default formats and data types, and how to change the defaults to suit your needs.

Using SAS Date and Time Values

The SAS System represents date, time and datetime values as numbers using the following rules:

- *A date is represented by the number of days between January 1, 1960 and that date.*
- *A time is represented as the number of seconds between midnight and that time of day.*
- *A datetime is represented by the number of seconds between midnight, January 1, 1960 and that datetime.*

The SAS System has a number of informats that read date and time values and convert them to SAS date and time values. The SAS System also has a number of formats that write date and time in a variety of ways.

SAS dates are valid back to A.D. 1582 and ahead to A.D. 20,000. Leap year, century, and fourth-century adjustments are handled correctly. However, leap seconds are ignored, and the SAS System does not adjust for daylight savings time.

You can create a SAS date constant or a SAS time constant by writing the date or time enclosed in single or double quotes, followed by a D (date), T (time), or DT (datetime) to indicate the type of value. Using the following patterns to create date and time constants:

'ddmmm<yy>yy'D
"ddmmm<yy>yy"D
represent a SAS date value. Refer to the following examples:

- *date='1jan1997'd;*
- *date='01jan96'd;*

'hh:mm<:ss.s>'T
"hh:mm<:ss.s>"T
represent a SAS time value. Refer to the following examples:

- *time='9:25't;*
- *time='9:25:19't;*

'ddmmm<yy>yy hh:mm<:ss.s>'DT
"ddmmm<yy>yy hh:mm<:ss.s>"DT
represent a SAS datetime value. Refer to the following examples:

- *dtime='18jan97:9:27:05'dt;*
- *if begin='01may97:9:30:00'dt then*
  end='31dec97:5:00:00'dt;

Refer to the following example, on how SAS stores date, time and datetime values.

```
mydate = '31MAR1960'D;

/* Since SAS stores date values as the number of days between January 1, */
/* 1960 and the date in question, we know that the variable mydate will be */
/* numeric and have a value of 90. */

     mytime = '00:15:00'T;

/* Since SAS stores time values as the number of seconds between midnight */
/* and the time in question, we know that the variable mytime will be numeric */
/* and have a value of 900. */
```

```
mydatetm = '02JAN1960:01:00:00'DT;

/* Since SAS stores datetime values as the number of seconds between      */
/* midnight on January 1, 1960 and the datetime in question, we know that */
/* the variable mydatetm will be numeric  and have a value of 90,000      */
/* (86,400 seconds in a day + 3,600 in one hour).                         */

put mydate = mytime= mydatetm= ;
run;

/* SAS LOG Result: MYDATE=90 MYTIME=900 MYDATETM=90000    */

/* We can use SAS formats to instruct SAS to display these values in a form */
/* that is more easily understood.  Keep in mind that the format does not   */
/* have any effect on the value stored in the data set, it only affects how the */
/* value is printed to output by the various SAS procedures and statements.  */

data b;
  set a;
  format mydate mmddyy10.;
  format mytime time10.2;
  format mydatetm datetime20.2;
  put mydate= mytime= mydatetm= ;
run;

/* SAS LOG Result: MYDATE=03/31/1960
                   MYTIME=0:15:00.00
                   MYDATETM=02JAN60:01:00:00.00          */

/* It is important to note that there is nothing special about these time, date,  */
/* or datetime variables.  They are simply numeric variables to SAS.              */
/* The value 900 has no intrinsic meaning absent some indicator to tell SAS       */
/* what the 900 means. Oracle uses the column type attribute, SAS uses the        */
/* format. Thus the variable mytime from the example above (mytime=900)           */
/* can mean several different things depending upon which format is               */
/* associated with it.  If you print it with a time format, it means 12:15 A.M.   */
/* If you print it with a date format, it means June 16, 1962.  If you print it   */
/* with a datetime format, it means 12:15 A.M. on January 1, 1960.  This is       */
/* shown in the example below and serves to illustrate the importance of          */
/* format statements and type statements when loading SAS data into               */
/* databases systems.                                                             */
```

```
data c;
  set a;
  put mytime time8.;
  put mytime mmddyy8.;
  put mytime datetime18.;
run;

/* SAS LOG Result:  0:15:00
                    06/19/62
                    01JAN60:00:15:00                    */
```

For more information on SAS date and time values, refer to the manual,"SAS Language Reference, Version 6".

PROC DBLOAD

The DBLOAD Procedure converts the SAS date variable formats into DB2 data types by default, as follows:

SAS Variable Format	*DB2 Data Type*
DATETIMEw.d	*TIMESTAMP*
TODw.	*TIMESTAMP*
*DATEw.**	*DATE*
*TIMEw.d***	*TIME*

**Includes all SAS date formats, such as MMDDYYw., JULIANw. and QTRw., etc.*
***Included all SAS time formats, such as HHMMw.d, MMSSw.d, and HOURw.d.*

USING THE DEFAULT DB2 DATA TYPES

The example below demonstrates how to load a SAS data set containing different SAS date formats into a DB2 table using PROC DBLOAD with the default DB2 data types. PROC DBLOAD passes the SAS format to DB2 and determines from the format what default data type to use. Therefore, if the FORMAT statement below was not used then SAS used thee the default format would be numeric (in this case, `BEST`*w., if the datetime, date and time formats were not used).*

```
data dataset;
  input  @1 date1 datetime18. @20 date2 date7. @28 date3 time8.;
  format date1 datetime18. date2 date7. date3 time8.;
cards;
08JUL1996:16:24:43 08JUL96 16:24:43
12SEP1964:14:22:00 12SEP64 14:22:00
30OCT1996:08:02:11 30OCT96 08:02:11
;
proc dbload dbms=db2_unix data=dataset;
  user="usr_name";
  using="password";
  database=db_name;
  table=datetest;
  load; run;
```

The following results are generated using the DB2 Command Line Processor.

```
db2 => select * from datetest

DATE1                      DATE2      DATE3
-------------------------- ---------- --------
1996-10-30-08.02.11.000000 10/30/1908 08:02:11
1964-09-12-14.22.00.000000 09/12/1914 14:22:00
1996-07-08-16.24.43.000000 07/08/1916 16:24:43
  3 record(s) selected.
```

******* DATE1 *is defined as DB2 data type* `timestamp`
******* DATE2 *is defined as DB2 data type* `date`
******* DATE3 *is defined as DB2 data type* `time`

```
db2 => select tbname, name, coltype, length from
       sysibm.syscolumns where tbname='DATETEST'

TBNAME              NAME                COLTYPE  LENGTH
------------------- ------------------- -------- ------
DATETST1            DATE1               TIMESTMP     10
DATETST1            DATE2               DATE          4
DATETST1            DATE3               TIME          3

  3 record(s) selected.
```

OVERRIDING THE DEFAULT DB2 DATA TYPES

Using the TYPE statement with PROC DBLOAD provides the capability of changing the default DB2 data types in the new table. The example below creates a new table and overrides the default DB2 data type. In this case, the FORMAT statement was not used, therefore SAS used the default format `BEST`*w. To create a table with the* `timestamp`, `date` *and* `time` *DB2 data types, you *must* use PROC DBLOAD with the TYPE statement as follows:*

```
data dataset;
  input  @1 date1 datetime18. @20 date2 date7. @28 date3 time8.;
cards;
08JUL1996:16:24:43 08JUL96 16:24:43
12SEP1964:14:22:00 12SEP64 14:22:00
30OCT1996:08:02:11 30OCT96 08:02:11
;
proc dbload dbms=db2_unix data=dataset;
  user="usr_name";
  using="password";
  database=db_name;
  table=datetest;
  type 1="TIMESTAMP" 2="DATE" 3="TIME";
  load;
run;
```

In this example, notice that FORMAT statements were NOT used when creating the SAS dataset, hence the default SAS data types are `numeric`. *If the TYPE statements were not used in PROC DBLOAD, then the DB2 default data type would be* `float`. *However, since the TYPE statements are used, the default SAS variable formats were overridden. The DB2 command line processor results are EXACTLY the same as the previous example.*

```
db2 => select * from datetest

DATE1                      DATE2      DATE3
-------------------------- ---------- --------
1996-10-30-08.02.11.000000 10/30/1908 08:02:11
1964-09-12-14.22.00.000000 09/12/1914 14:22:00
1996-07-08-16.24.43.000000 07/08/1916 16:24:43
  3 record(s) selected.
```

******* DATE1 *is defined as DB2 data type* `timestamp`
******* DATE2 *is defined as DB2 data type* `date`
******* DATE3 *is defined as DB2 data type* `time`

PROC SQL Pass-Through

By default, the SQL Procedure Pass-Through Facility returns the following DB2 data types as follows:

DB2 Data Type	*SAS Variable Format*
DATE	*DATE9.*
TIME	*TIME11.2*
TIMESTAMP	*DATETIMEw.d*

OVERRIDING THE DEFAULT DB2 DATA TYPES

If you want to override the default formats, you can modify the results by writing PROC SQL code similar to the following:

```
proc sql;
connect to db2 (user=usr_name using=password database=db_name);
create table sas.dataset as
     select date1 format=datetime7.,
            date2 format=worddate10.,
            date3 format=hour8.
     from connection to db2 (select * from datetest);
disconnect from db2;
quit;
```

PROC PRINT would generate the following report:

```
OBS      DATE1          DATE2       DATE3
 1     08JUL96    July                 16
 2     12SEP64    September            14
 3     30OCT96    October               8
```

You can also use the datepart() and timepart() functions to store the value that is needed so the default format is overridden, as shown below:

```
proc sql;
connect to db2 (user=usr_name using=password database=db_name);
  create table sas.dataset as
    select * from connection to db2
      (select * from datetest);
disconnect from db2;
quit;

data timetest;
 set sas.dataset;
 date1=timepart(date1);
 format date1 time8.;
run;
```

PROC PRINT would generate the following report:

```
OBS       DATE1          DATE2                 DATE3
 1     16:24:43    08JUL96:00:00:00           59083
 2     14:22:00    12SEP64:00:00:00           51720
 3      8:02:11    30OCT96:00:00:00           28931
```

Using PROC SQL Views in a SAS Data Step

Using SQL views that are created using PROC SQL Pass-Through to process date and times can be difficult to comprehend.

Note: PROC SQL VIews are READ ONLY (DBMS tables cannot be updated).
PROC ACCESS Views are READ and WRITE (DBMS tables can be updated).

SAS/ACCESS Interface to DB2/6000 provides the capability to access DB2 tables using SQL views. The following samples were tested with PROC SQL Pass-Through Views.

A WHERE statement on a data step that refers to an SQL view can be used. However, if the WHERE clause conforms to the DBMS syntax then the WHERE clause will be passed directly to the DBMS for processing. For instance:

Consider the SQL view MYVIEW that points to a DB2 table and references a column `date1`*, which is of data type* `TIMESTAMP` *in DB2. Remember that the DB2 data type* `TIMESTAMP` *is equivalent to the SAS data type* `DATETIME`*.*

```
data timetest;
 set MYVIEW;
 where date1 > '01JAN95'D;
```

Since you are attempting to use a SAS `DATE` *literal on the WHERE clause and the DB2 Server is expecting a* `TIMESTAMP` *value, the query will not be CORRECT. The WHERE clause* `DATE` *literal '01JAN95'D is represented by the number of days between January 1, 1960 and that date. The DB2 Server* `TIMESTAMP` *is expecting a character string representation that starts with a digit and has a length of at least 16 and has the form yyyy-mm-dd-hh.mm.ss.nnnnnn. Therefore, the above query will send back incorrect results.*

If the WHERE clause does NOT conform to the DBMS syntax then SAS will not send this WHERE clause to the DBMS for processing and instead, will ask the DBMS to send over ALL the data and SAS will discard those rows that do not meet the criteria.

Now consider:

```
data timetest;
 set MYVIEW;
 where date1 > '01JAN95:00:00:00'DT;
```

Since both the literal on the WHERE clause and the column in DB2 are of matching types and the WHERE clause conforms to the DBMS syntax, this WHERE clause will be passed along to DB2 for processing with all corresponding performance benefits.

Hints: If you use SAS Data Step Functions or SAS specific operators on the WHERE clause, the WHERE clause will not be passed to the DBMS.

If the SQL view was defined with a WHERE clause, the WHERE clause on the data step will be ANDED with it.

LIBNAME Engine ACCESS Support

This new LIBNAME engine ACCESS support is only available with SAS Release 7.x. In V7 SAS, a user can assign a LIBNAME statement to a "view" or "schema" of a DBMS database and a connection can be made when the LIBNAME statement is executed. Every use of a two-level SAS name, e.g. MYLIB.DATETEST, will cause the DBMS engine to dynamically read the DBMS DATETEST metadata. Both reading DBMS data and writing DBMS data will be supported using either DATA step or any of the PROCS. This is possible because of Dynamic Libnames.
The LIBNAME engine support, by default, converts the SAS date variable formats into DB2 data types the same as PROC DBLOAD (refer to PROC DBLOAD date conversion chart). It returns the same DB2 data types as the PROC SQL Pass-Through Facility (refer to the PROC SQL Pass-Through Facility date chart).

USING THE DEFAULT DB2 DATA TYPES

The example below demonstrates how to load a SAS data set containing different SAS date formats into a DB2 table , `datetest`*, using the LIBNAME engine with the default DB2 data types. The LIBNAME engine for ACCESS passes the SAS format to DB2 and determines from the format what default data type to use. Therefore, if the FORMAT statement below was not used then the default format would be numeric (in this case, E12., if the datetime, date and time formats were not used).*

```
libname DB2_CNCT DB2_UNIX USER="user_name" USING="password";

data dataset;
  input  @1 date1 datetime18. @20 date2 date7. @28 date3 time8.;
  format date1 datetime18. date2 date7. date3 time8.;
cards;
08JUL1996:16:24:43 08JUL96 16:24:43
12SEP1964:14:22:00 12SEP64 14:22:00
30OCT1996:08:02:11 30OCT96 08:02:11
;

data DB2_CNCT.datetest;
set dataset;
run;
```

The following results are generated using the DB2 Command Line Processor.

```
db2 => select * from datetest

DATE1                      DATE2      DATE3
-------------------------- ---------- --------
1996-10-30-08.02.11.000000 10/30/1908 08:02:11
1964-09-12-14.22.00.000000 09/12/1914 14:22:00
1996-07-08-16.24.43.000000 07/08/1916 16:24:43
  3 record(s) selected.
```

******* DATE1 *is defined as DB2 data type* `timestamp`
******* DATE2 *is defined as DB2 data type* `date`
******* DATE3 *is defined as DB2 data type* `time`

```
db2 => select tbname, name, coltype, length from
       sysibm.syscolumns where tbname='DATETEST'

TBNAME              NAME                COLTYPE  LENGTH
------------------- ------------------- -------- ------
DATETEST            DATE1               TIMESTMP     10
DATETEST            DATE2               DATE          4
DATETEST            DATE3               TIME          3

  3 record(s) selected.
```

OVERRIDING THE DEFAULT DB2 DATA TYPES

If you want to modify the default formats and informats that are returned to SAS from DB2/6000, you can modify the results similar to the following:

```
libname DB2_CNCT DB2_UNIX USER="user_name" USING="password";

proc print data=DB2_CNCT.datetest;
format date1 datetime7. date2 worddate10. date3 hour8.;
run;

-OR-

data mod_date;
set DB2_CNCT.datetest;
format date1 datetime7. date2 worddate10. date3 hour8.;
run;

proc print data=mod_date;
run;
```

Using SAS Dates with MACROS

Processing dates using the MACRO facility has caused some confusion and misunderstandings. Below are some examples with PROC SQL Pass-Through Facility, PROC DBLOAD and the LIBNAME engine support, that should help clarify the process.

The following example demonstrates the use of two DBMS dates to restrict the query using the Proc SQL Pass-Through Facility.

```
/* If creating start and end macros within a datastep...   */

call symput ("start", '" || put(strtdate, datetime18.) || "');
call symput ("end",   '" || put(enddate,  datetime18.) || "');

-OR-

/* If creating start and end macros with %let...           */

%let dset=dataset;
%let start=19960601000000;
%let start=%nrbquote(')&start%nrbquote(');

%let end=19970601000000;
%let end=%nrbquote(')&end%nrbquote(');

%let incond=where date1 between
                  timestamp(&start)
                  and
                  timestamp(&end);

libname sas '/Fully/Qualified/Path';

proc sql;
  connect to db2_unix( user=usr_id using=password
        database=db_name);
    create table sas.&dset as
       select date1 format datetime16.,
              date2 format date7.,
              date3 format time8.
       from connection to db2_unix
         (select date1 as d1,
                 date2 as d2,
                 date3 as d3
          from DATETEST
          &incond);
  disconnect from db2_unix;
quit;
```

PROC PRINT would generate the following report:

```
OBS                   D1          D2           D3
 1     08JUL96:16:24:43    08JUL96    16:24:43
 2     30OCT96:08:02:11    30OCT96     8:02:11
```

The example below demonstrates the use of a macro to process dates using a WHERE clause in a SAS data step using an SQL view.

```
data _null_;
input datelim datetime16.;
call symput ('wherdate', put(datelim,datetime16.) );
cards;
01jan95:00:00:00
;
run;

%put &wherdate;

data timetest;
  set myview;      /* SQL VIEW of DATETEST table */
     where date1 > "&wherdate"dt;
run;
```

The next example demonstrates the use of macros to retrieve dates from a DBMS table using the LIBNAME engine support, available at SAS V7.

```
libname DB2_CNCT DB2 USER="user" USING="password";
libname sas '/Fully/Qualified/Path';

%let dset=sas.dataset;

%let start=19960601000000;
%let start=%nrbquote(')&start%nrbquote(');

%let end=19970601000000;
%let end=%nrbquote(')&end%nrbquote(');

%let incond="where date1 between
             timestamp(&start)and timestamp(&end)";

 data &dset;
  set DB2_CNCT.datetest ( dbcondition=&incond );
  format date1 datetime16. date2 date7. date3 time8.;
 run;

proc print data= &dset;
run;
```

PROC PRINT would generate the following report:

```
OBS         DATE1              DATE2       DATE3
 1      08JUL96:16:24:43     08JUL96     16:24:43
 2      30OCT96:08:02:11     30OCT96      8:02:11
```

Valid SAS FORMATS for Overriding the Default DATE/TIME FORMATS

DB2 data type	*Valid SAS formats*
TIMESTAMP	*DATETIMEw.d *ONLY**
	*** Time Formats return valid time values which are not accurate times compared with original timestamp because DB2 internally stores a timestamp differently than what SAS requires. Date Formats do not return any value. Therefore, all Time and Date formats are not supported for overriding default DATE/TIME formats.*
DATE	*DATEw., DDMMYYw., DAYw., JULDAYw., JULIANw., MMDDYYw., MMYYxw., MONNAMEw., MONTHw., MONYYw., WEEKDATEw., WEEKDATXw., WEEKDAYw., WORDDATEw., WORDDATXw., YEARw., YYMMxw., YYMMDDw., YYMONw., YYQxw., YYQRxw.*
	*** Datetime and Time Formats return values which are not accurate datetimes and times compared with original date and time because DB2 internally stores a timestamp differently than what SAS requires; therefore, they are not supported for overriding default DATE/TIME formats.*
TIME	*HHMMw.d, HOURw.d, MMSSw.d, TIMEw., TODw.*
	*** Datetime and Date Formats return values which are not accurate datetimes and dates because the time values in DB2 do not contain any knowledge of a date; therefore, all Datetime and Date Formats are not supported for overriding default DATE/TIME formats.*

NOTE: Informats are accepted but are ignored using PROC SQL Pass-Through

Conclusion

This paper discussed the various aspects of processing dates with SAS/ACCESS Interface to DB2/6000 on UNIX platforms. This discussion included date processing using the SQL Procedure Pass-Through Facility, the DBLOAD Procedure and the LIBNAME engine support available with SAS V7 for DB2/6000. It also included specific issues involving date formats for DB2/6000 and using DB2/6000 dates with MACROS.

For detailed information on the SAS procedures PROC DBLOAD, PROC SQL PASS-THROUGH and the LIBNAME engine support for ACCESS, reference to the following manuals:

SAS/ACCESS Software for Relational Databases Reference, Version 6

SAS/ACCESS Software for Relational Databases Reference, Version 7

SAS/ACCESS Software Changes and Enhancements, SQL Procedure Pass-Through Facility, Version 6

SAS Software Changes and Enhancements, Release 6.11

Getting Started with SAS/ACCESS Software, Version 6

SAS Language Reference, Version 6

For detailed information on the DBMS data types or functions used with the examples in this report, please refer to your Database Administrator and your DBMS manuals.