DFSORT upgrade with some very useful conversion/formatting/correcting features

by George Lewycky TIS-Financial Systems
george.lewycky@nyct.com

Recently I found and requested upgrades to IBM for a useful feature that is available with DFSORT. The latest upgrade (**on our "C", "D" and "E" LPAR's**) allows DFSORT to unpack signed numerical data (particularly dollar amounts!) into a format containing the full numeric value, negative sign (if applicable) and the cents with a decimal point.

NEW: Also you can convert (correct) numeric packed fields containing low-values and zero-out the field for data transfer, conversions, etc. (see pages 14-19 for more info)

With most of us having to both, port and convert data from the mainframe's EBCDIC to various platforms (Unix and Windows) which require ASCII, this features speeds up the process especially for ad-hoc, FTP's, special requests and one-time requests.

Also with spreadsheets, we need this functionality and without needing to code for it especially field-by-field. This process is much simpler and quicker.

COBOL or SAS programs are usually used by most of us for this but now with the latest release of DFSORT that I requested and tested **on our "C", "D" and "E" boxes (LPAR's)** COBOL and SAS are not needed anymore except for complex logic, VSAM, etc.

IMPORTANT NOTE(s):

LPARS:

Only remaining LPAR that doesn't have these DFSORT upgrades is the AFC machine. AFC management didn't request for this.

"E" LPAR ONLY:

Only class "X" is setup with initiators on the "E" box, so be sure to change your class in your JCL before testing if using the "E" box

ALL LPARS:

STEPLIB overrides are not needed for this version of DFSORT

Later in this document I will illustrate how this done with an example but I need to highlight a few features and give some references.

Play close attention to the **EDIT Sub parameter and Edit Masks sections (in Table 38 on page 5)** below.

Or you can create you own as I illustrate on page 11.

References & Web links:

IBM's Main DFSORT/MVS
http://www.ibm.com/storage/dfsort/
OR
http://www-03.ibm.com/servers/storage/support/software/sort/mvs/index.html

Table 3.13 OUTFIL Control Statements (see page 4 of this document) :
http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/iceca109/3.13
Be sure to look at **IBM's 27 predefined** edit masks in **Table 38** in this link, especially before you create your own.

DFSORT Documentation:
http://www-03.ibm.com/servers/storage/support/software/sort/mvs/srtmpub.html

Title: DFSORT R14 Application Programming Guide **Document Number:** SC33-4035-21
http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/iceca109/CCONTENTS

Other Links:

Ask Professor Sort:
http://www-03.ibm.com/servers/storage/support/software/sort/mvs/professor_sort/index.html

Beyond Sorting:
http://www-03.ibm.com/servers/storage/support/software/sort/mvs/beyond_sorting/online/index.html

Smart DFSORT Tricks:
http://www-03.ibm.com/servers/storage/support/software/sort/mvs/tricks/index.html

IBMMAINFRAMES.com Expert Forum -> VSAM & DFSORT**:**
http://ibmmainframes.com/forum/viewforum.php?f=8
NOTE: This is moderated by **Frank Yaeger (DFSORT guru)**

DFSORT/MVS : Reformatting and OUTFIL Features:
http://www-03.ibm.com/servers/storage/support/software/sort/mvs/beyond_sorting/online/srtmboft.html

19.5.6 OUTFIL numeric editing

OUTFIL OUTREC provides sophisticated editing capabilities for controlling how numeric fields, including SMF date and time fields and two-digit year date fields, are presented with respect to length, leading or suppressed zeros, thousands separators, decimal points, leading and trailing positive and negative signs, and so on. You can edit BI, FI, PD, PD0, ZD, CSF/FS, Y2x, DT1, DT2, DT3, TM1, TM2, TM3 and TM4 format fields. Twenty-seven pre-defined editing masks are available for commonly used numeric patterns, encompassing many of the numeric notations used throughout the world. In addition, a virtually unlimited number of numeric editing patterns are available via user-defined editing masks.

OUTREC's editing capabilities are particularly useful for OUTFIL reports. Here is an example of OUTFIL numeric editing:

```
OUTFIL FNAMES=(OUT1,BACKUP),OUTREC=(5:21,7,ZD,M18,
          25:46,3,ZD,M12,
          51:8,3,PD,EDIT=(IT.TTT))
```

The output records for the OUT1 and BACKUP data sets will contain the following:

_ Blanks in output positions 1-4.

_ In output positions 5-14, the ZD value in input positions 21-27 converted to a pattern of SII,IIT.TT where S represents a sign of blank for plus or - for minus, I represents a leading insignificant digit and T represents a significant digit.

_ Blanks in output positions 15-24.

_ In output positions 25-29, the ZD value in input positions 46-48 converted to a pattern of ST.TT where S represents a sign of blank for plus or - for minus and T represents a significant digit.

_ Blanks in output positions 30-50.

_ In output positions 51-56, the PD value in input positions 8-10 converted to a pattern of IT.TTT where I represents a leading insignificant digit and T represents a significant digit.

The OUTREC and INREC statements can also be used to edit numeric input fields using the same capabilities available for OUTFIL OUTREC. Here's the same example above using OUTREC:

```
OUTREC FIELDS=(5:21,7,ZD,M18,25:46,3,ZD,M12,
51:8,3,PD,EDIT=(IT.TTT))
```

From:

Title: DFSORT R14 Application Programming Guide
Document Number: SC33-4035-21

http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/iceca109/3.13

| Mn specifies one of twenty-seven pre-defined edit masks
| (M0-M26) for presenting numeric data. If these pre-defined edit masks are not suitable for presenting your numeric data, the EDIT parameter gives you the flexibility to define your own edit patterns.

| The twenty-seven pre-defined edit masks can be represented
| as follows:

Table 38. Edit Mask Patterns

Mask	Pattern	Examples	
		Value	Result
M0	IIIIIIIIIIIIIITS	+01234	1234
default		-00001	1-
M1	TTTTTTTTTTTTTTTS	-00123	00123-
		+00123	00123
M2	I,III,III,III,IIT.TTS	+123450	1,234.50
		-000020	0.20-
M3	I,III,III,III,IIT.TTCR	-001234	12.34CR
		+123456	1,234.56
M4	SI,III,III,III,IIT.TT	+0123456	+1,234.56
		-1234567	-12,345.67
M5	SI,III,III,III,IIT.TTS	-001234	(12.34)
		+123450	1,234.50
M6	III-TTT-TTTT	00123456	012-3456
		12345678	1-234-56788
M7	TTT-TT-TTTT	00123456	000-12-3456
		12345678	012-34-5678
M8	IT:TT:TT	030553	3:05:53
		121736	12:17:36
M9	IT/TT/TT	123094	12/30/94
		083194	8/31/94
M10	IIIIIIIIIIIIIIT	01234	1234
		00000	0

M11	TTTTTTTTTTTTTTT	00010	00010
		01234	01234
M12	SIII,III,III,III,IIT	+1234567	1,234,567
		-0012345	-12,345
M13	SIII.III.III.III.IIT	+1234567	1.234.567
		-0012345	-12.345
M14	SIII III III III IITS	+1234567	1 234 567
		-0012345	(12 345)
M15	III III III III IITS	+1234567	1 234 567
		-0012345	12 345-
M16	SIII III III III IIT	+1234567	1 234 567
		-0012345	-12 345
M17	SIII'III'III'III'IIT	+1234567	1'234'567
		-0012345	-12'345
M18	SI,III,III,III,IIT.TT	+0123456	1,234.56
		-1234567	-12,345.67
M19	SI.III.III.III.IIT,TT	+0123456	1.234,56
		-1234567	-12.345,67
M20	SI III III III IIT,TTS	+0123456	1 234,56
		-1234567	(12 345,67)
M21	I III III III IIT,TTS	+0123456	1 234,567
		-1234567	12 345,67-
M22	SI III III III IIT,TT	+0123456	1 234,56
		-1234567	-12 345,67
M23	SI'III'III'III'IIT.TT	+0123456	1'234.56
		-1234567	-12'345.67
M24	SI'III'III'III'IIT,TT	+0123456	1'234,56
		-1234567	-12'345,67
M25	SIIIIIIIIIIIIIT	+01234	1234
		-00001	-1
M26	STTTTTTTTTTTTTT	1234	+01234
		-1	-00001

The elements used in the representation of the edit masks **(or user created Masks)** in Table 38 are as follows:

- **I** indicates a leading insignificant digit. **If zero, this digit will not be shown.**

- **T** indicates a significant digit. **If zero, this digit will be shown.**

- **CR (in M3)** is printed to the right of the digits if the value is negative; otherwise, two blanks are printed to the right of the digits.

- **S indicates a sign**.
  If it appears as the first character in the pattern, it is a **leading sign**.

  If it appears as the last character in the pattern, it is a **trailing sign**.

  If S appears as both the first and last characters in a pattern (example: M5), the first character is a leading sign and the last character is a trailing sign. Four different sign values are used: leading positive sign (lp), leading negative sign (ln), trailing positive sign (tp) and trailing negative sign (tn).

EDIT Subparameter

The EDIT subparameter provides the user with the capability of creating individual patterns for editing converted numeric data. An individual character pattern is composed of significant digit selectors, leading insignificant digit selectors, a sign replacement character, and all other characters that are to be printed in the actual output. The edit pattern may be up to 22 characters in length with a maximum of 15 leading insignificant and/or significant digits.

The characters used to represent significant or insignificant digit selectors are determined by the keyword EDxy. If EDIT is specified, the letter I is used to represent a leading insignificant digit which will print as a blank if the digit is zero, or the letter T is used to represent any significant digits (digits that are to be printed in their true form, even as leading zeros).

When a blank, quotation mark, or unbalanced parenthesis appears within an EDxy pattern, the pattern must be enclosed in single quotation marks. Balanced parentheses need not be enclosed within quotation marks. A single quotation mark within the pattern (for example, an apostrophe) must be specified as a double quotation mark.

The following illustrates the ability to replace the insignificant and significant digit selector characters x and y with any other characters:

```
+-------------------------------------------------------------------------+
|                    EDxy=                                                |
|                                                                         |
|            where  x = insignificant digit selector character           |
|                   y = significant digit selector character              |
+-------------------------------------------------------------------------+
```

All other characters are printed as specified in the edit pattern with the following exceptions:

- Any character specified after the first leading insignificant digit selector and before the first significant digit selector will be printed as a blank unless a previously selected digit was non-zero.

- Any character specified after the last significant digit selector will be printed as a blank if the edited number is positive.

- Any character or character string specified before the first leading insignificant digit selector, including a leading sign character if specified, will be printed to the immediate left of the first significant digit. The appropriate number of leading blanks will be supplied, assuring that the total number of characters in the printed field corresponds to the total number of characters in the edit pattern.

- Any leading insignificant digit selector specified after the first significant digit selector will be treated as a significant digit selector.

- The sign replacement character appearing as the first and/or last character of the pattern is replaced as per the SIGNS parameter.

Editing Masks

The editing masks are provided in order to simplify the more common editing operations. If neither "Mm" nor EDIT are specified in the OUTREC control statement, the default mask, M0, is used.

The following chart illustrates the SyncSort-supplied editing masks. Leading insignificant digits are represented by the letter I and significant digits are represented by the letter T. Leading or trailing sign replacement characters are represented by the letter S. All other characters print as they appear in the pattern.

The Signs illustrated for each mask are formatted in conjunction with the

format requirements for the SIGNS parameter. The user may specify the SIGNS parameter to selectively override the signs for a particular mask. For example, if the user specifies M4 and also specifies SIGNS=(' '), Syncsort will print a leading blank instead of a plus sign if the number is positive. Syncsort will print a leading minus sign if the number is negative because the leading negative sign specified in the editing mask has not been overriden.

The LENGTH value formulas give the length, in bytes, as a result of the editing operation as a function of the number of input digits (d). The value for d can be determined by consulting the Data Conversion table which follows the Editing Mask chart. D is a function of the input field format and length.

In the Editing Mask chart, the half bracket symbol | | indicates that only the integer part of the division be retained. + +

Editing Masks

Mask	Pattern	Signs	Length
M0 -- Default	IIII...IITS	(,,' ',-)	d+1
M1	TTTTT...TTTS	(,,' ',-)	d+1
M2	I,III,...,IIT.TTS	(,,' ',-)	d+2 + | d-3 | / 3 +
M3	I,III,...,IIT.TTCR		d+3 + | d-3 | / 3 +
M4	SI,III,...,IT.TT	(+,-)	d+2 + | d-3 | / 3 +
M5	SI,III,...,IIT.TTS	(' ',(' ',))	d+3 + | d-3 | / 3 +
M6	III-TTT-TTTT		12
M7	TTT-TT-TTTT		11
M8	IT:TT:TT		8
M9	IT/TT/TT		8

Data Conversion

Input Format	Number of Bytes in Input Field	Number of Resulting Digits (d)
ZD	1	1
PD	1	2l-1
BI	1	3
BI,FI	2	5
BI	3	8
BI,FI	4	10

Miscellaneous

One problem I encountered was data that mismatched the field (**see example below**) which resulted with an S0C7. This can be prevalent if you have an old system or numerous feeds.

After tracing the records and then omitting them the conversion went perfect.

So, if you encounter this problem it might be data related and not your JCL or sort cards.

You will have to omit these records, values, etc. before you use this feature otherwise you will encounter S0C7 abends and/or DFSORT errors.

```
05 VLJ-AMTS-AREA                    289
   10 VLJ-AMTS(1)                   289 X'0CF0C040F0404040'
   10 VLJ-AMTS(2)                   297 4040404040000.00
```

This additional JCL will help you validate your data and/or find any invalid values in the field you are converting:

```
//S1     EXEC  PGM=ICETOOL
//TOOLMSG    DD  SYSOUT=*
//DFSMSG     DD  SYSOUT=*
//IN DD DSN=input file
//TOOLIN DD *
VERIFY FROM(IN) ON(289,8,PD)
/*
```

Explanation of the TOOLIN dd card:

ON(289,8,PD) ← this designates the starting column & length of the data that is packed that you want to verify.

For a ZD or PD format field:

An invalid digit results in a data exception (0C7 ABEND) or incorrect numeric output; A-F are invalid digits. ICETOOL's VERIFY or DISPLAY operator can be used to identify decimal values with invalid digits.

A value is treated as positive if its sign is F, E, C, A, 8, 6, 4, 2, or 0.
A value is treated as negative if its sign is D, B, 9, 7, 5, 3, or 1.

For more info on the ICETOOL's VERIFY look on page 8 at the link/pdf below and the others beneath it:

http://www-03.ibm.com/servers/storage/support/software/sort/mvs/icetool/pdf/sorttool.pdf

http://www-03.ibm.com/servers/storage/support/software/sort/mvs/professor_sort/srtmatol.html

http://ibmmainframes.com/post-4796.html

A STEP BY STEP EXAMPLE

Sample JCL:

```
//SYSIN DD *
 SORT FIELDS=(COPY)
 INCLUDE COND=(11,2,CH,EQ,C'LD',AND,
     289,12,CH,NE,X'0CF0C040F040404040404040')
 OUTREC FIELDS=(5X,289,8,PD,EDIT=(SIIIIIIIT.TT),SIGNS=(,-))
/*
```

NOTE:

- the 'edit mask' in bold is customized.
- For my use IBM didn't have a mask to suit my needs.
- IBM has 27 pre-defined edit masks (M0 – M26) or you can customize your own!

Sample record mapped with FILE-AID:

OFFSET:

```
RECORD:      1                        VLJ-RECORD
---- FIELD LEVEL/NAME ------- COLUMNS- ----+----1----+----2----+----3--
5 VLJ-DCCD-AREA                    287
  10 VLJ-DCCD(1)                   287 C
  10 VLJ-DCCD(2)                   288
5 VLJ-AMTS-AREA                    289
  10 VLJ-AMTS(1)                   289 -0000000002082.22
  10 VLJ-AMTS(2)                   297 000000000000000
```

FORMATTED:

```
5 VLJ-AMTS-AREA                GROUP
  10 VLJ-AMTS(1)             S9(13)V99 -0000000002082.22
```

EBCDIC data in hex before conversion	**After conversion**
00000022 0000282D	-2082.22
00000650 0000103D	-1605.30
00000598 0000306D	-3509.68
00000598 0000306C	3509.68

Sample JCL corresponding to the record/field shown above:

```
//SYSIN DD *
 SORT FIELDS=(COPY)
 INCLUDE COND=(1,4,CH,EQ,C'0VLJ',AND,11,2,CH,EQ,C'LD')
 OUTREC FIELDS=(5X,289,8,PD,EDIT=(SIIIIIIIT.TT),SIGNS=(,-))
/*
```

NOTE: you can alter the position of the sign bit (**S**) to the end of the numeric value as follows (see bottom of Page 6):

```
    OUTREC FIELDS=(5X,289,8,PD,EDIT=(IIIIIIIT.TTS),SIGNS=(,-))
```

FINALLY THIS IS THE END-RESULT:

Output file will look like:

```
----+----1----+--
          -651.86
          -405.54
         -3344.52
         -3034.08
         -5322.93
         -3445.10
         -2235.54
         -7106.65
```

NOTE:
If you DON'T want the zeros suppressed see the next page

NEW:

If you don't prefer ZERO suppression change the SORT CARD as follows:

```
FROM:  OUTREC FIELDS=(5X,289,8,PD,EDIT=(SIIIIIIIT.TT),SIGNS=(,-))
TO   :  OUTREC FIELDS=(5X,289,8,PD,EDIT=(STTTTTTTT.TT),SIGNS=(,-))
```

```
//SYSIN DD *
 SORT FIELDS=(COPY)
 INCLUDE COND=(1,4,CH,EQ,C'0VLJ',AND,11,2,CH,EQ,C'LD')
 OUTREC FIELDS=(5X,289,8,PD,EDIT=(STTTTTTTT.TT),SIGNS=(,-))
/*
```

Output file will look like:

```
-00000651.86
-00000405.54
-00003344.52
-00003034.08
-00005322.93
-00003445.10
-00002235.54
-00007106.65
```

"TO ZERO SUPRESS OR NOT ZERO SUPRESS"

Retaining the Zero's might be safer and beneficial for files when you need consistency and data in fixed columns (Such as, COBOL, FILE-AID and SAS programs) along with visual readability.

Contrary to Excel, or utilities such as Oracle's SQL*Loader which can be driven by unique delimiters along with fixed column positions and lengths. It can simply be a choice of preference or experience

Converting invalid numeric packed decimal (eg. low-values) to a valid numeric value (eg. zero)

From: Howard Schwartz TIS, Financial Systems
Howard.Schwartz@nyct.com

References:
http://ibmmainframes.com/forum/viewtopic.php?t=10683

http://www-03.ibm.com/servers/storage/support/software/sort/mvs/beyond_sorting/pdf/sortbynd.pdf (see p.15)

Now you can examine fields that contain invalid numeric data and replace with valid data using DFSORT also!!

In this example I am replacing (correcting) a COMP-3 field that contains invalid data and assigning +0 to the same field.

If you have multiple fields to inspect in the same record, you must use the HIT=NEXT option.

If your requirements entail more than the data cleanup (converting from low-values) with including tasks such as omitting/including records, reformatting, reporting, collapsing, etc. its suggested you do these in a separate SORT STEP!
Do not complicate the CONVERSION logic, this is also so you can review the converted file before the steps.

```
//SORT    EXEC PGM=UTIL,PARM=SORT
//SORTLIB   DD DSN=SYS1.SORTLIB,DISP=SHR
//SYSOUT    DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SORTIN    DD DSN=infile,DISP=SHR
//SORTOUT   DD DSN=outfile,DISP=SHR
//SYSIN DD *
SYSIN DD *
SORT FIELDS=COPY
 INREC IFTHEN=(WHEN=(298,6,PD,NE,NUM),
         OVERLAY=(298:+0,TO=PD,LENGTH=6),
           HIT=NEXT),
       IFTHEN=(WHEN=(304,6,PD,NE,NUM),
         OVERLAY=(304:+0,TO=PD,LENGTH=6))
```

Before sort:

```
TA1-DIRECT-AMOUNT
6/PS
(298-303)
120--------------

000000            ← LOW-VALUES
000000

INREC IFTHEN=(WHEN=(298,6,PD,NE,NUM),
        OVERLAY=(298:+0,TO=PD,LENGTH=6))
```

After sort:

```
TA1-DIRECT-AMOUNT
6/PS
(298-303)
120--------------

000000          ← VALID NUMERIC DATA OF +0
00000C
```

REFERENCES :

From:
http://www.mainserver.state.mn.us/bookmgr-cgi/bookmgr.exe/handheld/BOOKS/IBMBK.Z6MC.ICE1CM10.BOOK/CHANGES.1.1.2?SHELF=IBMBK.Z6MC.EZ2MZ601.BKSHELF&DT=20050119125222

CHANGES.1.1.2 OUTFIL Enhancements

| OUTFIL now allows you to reformat records in one of the following three
| ways using unedited, edited, or converted input fields and a variety of
| constants:

- | BUILD or OUTREC: The existing OUTREC parameter, or its new alias of
  | BUILD, allows you to reformat each record by specifying all of its
  | items one by one. BUILD or OUTREC gives you complete control over the
  | items you want in your reformatted OUTFIL records and the order in
  | which they appear. You can delete, rearrange and insert fields and
  | constants.
- | OVERLAY: The new OVERLAY parameter allows you to reformat each record
  | by specifying just the items that overlay specific columns. Overlay
  | lets you change specific existing columns without affecting the entire
  | record.
- | IFTHEN clauses: The new IFTHEN clauses allow you to reformat different
  | records in different ways by specifying how BUILD or OVERLAY items are
  | applied to records that meet given criteria. IFTHEN clauses let you
  | use sophisticated conditional logic to choose how different record
  | types are reformatted.

| OUTFIL OUTREC, as well as BUILD, OVERLAY, and IFTHEN, now allows you to
| use larger numeric values for fields and decimal constants to be edited,
| converted or used in arithmetic expressions. PD, ZD and FS fields, and
| decimal constants, can now be up to 31 digits. BI and FI fields can now be
| up to 8 bytes.

| OUTFIL OUTREC, as well as BUILD, OVERLAY, and IFTHEN, now allows you to
| use new UFF (unsigned free form) and SFF (signed free form) formats for
| fields to be edited, converted or used in arithmetic expressions.

| OUTFIL OUTREC, as well as BUILD, OVERLAY, and IFTHEN, now allows you to
| use new DC1-DC3 (TOD date), DE1-DE3 (ETOD date), TC1-TC4 (TOD time) and
| TE1-TE4 (ETOD time) formats for fields to be edited, converted or used in
| arithmetic expressions. These new formats produce meaningful
| representations of TOD and ETOD date and time values.

| OUTFIL OUTREC, as well as BUILD, OVERLAY, and IFTHEN, now allows you to
| use a set field in the CHANGE option (for example,
| 1,2,CHANGE=(4,C'FY',C'0001',C'VV',21,4)).

| OUTFIL OUTREC, as well as BUILD, OVERLAY and IFTHEN, now allows you to
| restart the sequence number when the binary value of a specified field
| changes (for example, SEQNUM,5,ZD,RESTART=(11,4)).

| OUTFIL OUTREC, as well as BUILD, OVERLAY, and IFTHEN, now allows you to
| use DATE, DATE=(abcd), DATENS=(abc), YDDD=(abc), YDDDNS=(ab), TIME,
| TIME=(abc) and TIMENS=(ab) options to insert the date and time of your
| DFSORT run into your records in various forms.

| OUTFIL OUTREC, as well as BUILD, OVERLAY, and IFTHEN, now allows you to
| use new TO=ZDC and TO=ZDF options to convert numeric values to ZD values
| with C or F for the positive sign, respectively. The TO=ZDF option is
| equivalent to the existing TO=ZD option.

| OUTFIL INCLUDE and OMIT now allow you to use larger FS values for compare
| fields. These values can now be up to 32 digits.

| OUTFIL INCLUDE and OMIT now allow you to use larger decimal constants for
| comparison to BI and FI fields. Decimal constants can now be up to
| +18446744073709551615 for comparison to BI fields. Decimal constants can
| now be between -9223372036854775808 and +9223372036854775807 for
| comparison to FI fields.

| OUTFIL INCLUDE and OMIT now allow you to use new UFF (unsigned free form)
| and SFF (signed free form) formats for compare fields. A UFF or SFF field
| can be compared to a UFF, SFF, FS, CSL or CST field or to a decimal
| constant.

| OUTFIL TRAILERx now allows you to use larger numeric values for
| statistical fields (total, maximum, minimum, average). PD, ZD and FS
| fields, and decimal constants, can now be up to 31 digits. BI and FI
| fields can now be up to 8 bytes.

| OUTFIL TRAILERx now allows you to use new UFF (unsigned free form) and SFF
| (signed free form) formats for statistical fields (total, maximum,minimum,
| average).

| OUTFIL TRAILERx now allows you to use TO=fo and fo (to) options to convert
| statistical fields (total, maximum, minimum, average) and counts to BI,
| FI, PD, ZD, ZDC, ZDF or FS output values.

| OUTFIL TRAILERx now allows you to use COUNT+n=(edit), COUNT+n=(to),
| COUNT-n=(edit) and COUNT-n=(to) to add or subtract n from a count to be
| edited or converted (for example, COUNT+1=(TO=ZD)).

| OUTFIL HEADERx and TRAILERx now allow you to insert hexadecimal strings
| (X'yy...yy' or nX'yy...yy') in your headers and trailers.

| OUTFIL HEADERx and TRAILERx now allow you to use new YDDD=(abc) and
| YDDDNS=(ab) options to insert the year (yyyy) and day of the year (ddd) of
| your DFSORT run in your headers and trailers.

| OUTFIL HEADERx and TRAILERx now allow you to use PAGE=(edit) and PAGE=(to)
| to edit or convert the page number (for example, PAGE=(M11,LENGTH=3)).

http://www-03.ibm.com/servers/storage/support/software/sort/mvs/summary_changes/srtmsocc.html

z/OS V1R5 and Release 14 - PTFs (April, 2006)

INCLUDE and OMIT Enhancements

COND now allows you to compare date fields in various formats to past and future dates (relative to the date of your DFSORT run) using new DATEn+r, DATEn-r, DATEn(c)+r, DATEn(c)-r, DATEnP+r, DATEnP-r, Y'DATEn'+r and Y'DATEn'-r constants. &DATEn+r, &DATEn-r, &DATEn(c)+r, &DATEn(c)-r, &DATEnP+r and &DATEnP-r can be used as aliases for DATEn+r, DATEn-r, DATEn(c)+r, DATEn(c)-r, DATEnP+r and DATEnP-r, respectively.

COND now allows you to test a field for numerics (field,EQ,NUM) or non-numerics (field,NE,NUM) in character (FS), zoned decimal (ZD) or packed decimal (PD) format.

COND now allows you to use &DATEn, &DATEn(c) and &DATEnP as aliases for DATEn, DATEn(c) and DATEnP, respectively.

INREC and OUTREC Enhancements

PARSE and IFTHEN PARSE are new options that allow you to extract variable position/length fields into fixed-length parsed fields defined as %nn fields. PARSE gives you powerful new capabilities for handling variable fields such as delimited fields, comma separated values (CSV), tab separated values, blank separated values, keyword separated fields, null-terminated strings, and many other types.

You can use various PARSE options to define the rules for extracting variable fields into up to one hundred %nn fixed-length parsed fields (%00-%99), and then use these %nn fields where you can use p,m fields in BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and FIELDS. You can edit, convert, justify, squeeze, translate, and do arithmetic with %nn fields.

BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and FIELDS now allow you to use a new JFY option to left-justify or right-justify the data in a field. For a left-justified field, leading blanks are removed and the characters from the first nonblank to the last nonblank are shifted left, with blanks inserted on the right if needed. For a right-justified field, trailing blanks are removed and the characters from the last nonblank to the first nonblank are shifted right, with blanks inserted on the left if needed.

Optionally for JFY, specific leading and trailing characters can be changed to blanks before justification begins, a leading string can be inserted, a trailing string can be inserted, and the output length can be changed.

BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and FIELDS now allow you to use a new SQZ option to left-squeeze or right-squeeze the data in a field. For a left-squeezed field, all blanks are removed and the characters from the first nonblank to the last nonblank are shifted left, with blanks inserted on the right if needed. For a right-squeezed field, all blanks are removed and the characters from the last nonblank to the first nonblank are shifted right, with blanks inserted on the left if needed.

Optionally for SQZ, specific characters can be changed to blanks before squeezing begins, a leading string can be inserted, a trailing string can be inserted, a string can be inserted wherever a group of blanks is removed between the first nonblank and the last nonblank, blanks can be kept as is between paired apostrophes or paired quotes, and the output length can be changed.

BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and FIELDS now allow you to insert past and future dates (relative to the date of your DFSORT run) into your records in various forms using new DATEn+r, DATEn-r, DATEn(c)+r, DATEn(c)-r, DATEnP+r and DATEnP-r constants. &DATEn+r, &DATEn-r, &DATEn(c)+r, &DATEn(c)-r, &DATEnP+r and &DATEnP-r can be used as aliases for DATEn+r, DATEn-r, DATEn(c)+r, DATEn(c)-r, DATEnP+r and DATEnP-r, respectively.

BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and FIELDS now allow you to use FL format to convert 4-byte or 8-byte hexadecimal floating-point values to integer values.

BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and FIELDS now allow you to use new TO=PDF and TO=PDC options to convert numeric values to PD values with F or C for the positive sign, respectively. The TO=PDC option is equivalent to the existing TO=PD option.

BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and FIELDS now allow you to use &DATEn, &DATEn(c), &DATEnP, &YDDD=(abc), &YDDDNS=(ab), &TIMEn, &TIMEn(c) and &TIMEnP as aliases for DATEn, DATEn(c), DATEnP, YDDD=(abc), YDDDNS=(ab), TIMEn, TIMEn(c) and TIMEnP, respectively.

IFTHEN WHEN now allows you to compare date fields in various formats to past and future dates (relative to the date of your DFSORT run) using new DATEn+r, DATEn-r, DATEn(c)+r, DATEn(c)-r, DATEnP+r, DATEnP-r,

Y'DATEn'+r and Y'DATEn'-r constants. &DATEn+r, &DATEn-r, &DATEn(c)+r, &DATEn(c)-r, &DATEnP+r and &DATEnP-r can be used as aliases for DATEn+r, DATEn-r, DATEn(c)+r, DATEn(c)-r, DATEnP+r and DATEnP-r, respectively.

IFTHEN WHEN now allows you to test a field for numerics (field,EQ,NUM) or non-numerics (field,NE,NUM) in character (FS), zoned decimal (ZD) or packed decimal (PD) format.

IFTHEN WHEN now allows you to use &DATEn, &DATEn(c) and &DATEnP as aliases for DATEn, DATEn(c) and DATEnP, respectively.

OUTFIL Enhancements

BLKCCH1 is a new report option that allows you to avoid forcing a page eject at the start of the report header; the ANSI carriage control character of '1' (page eject) in the first line of the report header (HEADER1) is replaced with a blank.

BLKCCH2 is a new report option that allows you to avoid forcing a page eject at the start of the first page header; the ANSI carriage control character of '1' (page eject) in the first line of the first page header (HEADER2) is replaced with a blank.

BLKCCT1 is a new report option that allows you to avoid forcing a page eject at the start of the report trailer; the ANSI carriage control character of '1' (page eject) in the first line of the report trailer (TRAILER1) is replaced with a blank.

SPLIT1R is a new option that allows you to write contiguous groups of records in one rotation among multiple output data sets. A specified number of records is written to each output data set and extra records are written to the last output data set.

PARSE and IFTHEN PARSE are new options that allow you to extract variable position/length fields into fixed-length parsed fields defined as %nn fields. PARSE gives you powerful new capabilities for handling variable fields such as delimited fields, comma separated values (CSV), tab separated values, blank separated values, keyword separated fields, null-terminated strings, and many other types.

You can use various PARSE options to define the rules for extracting variable fields into up to one hundred %nn fixed-length parsed fields (%00-%99), and then use these %nn fields where you can use p,m fields in BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and OUTREC. You can edit, convert, justify, squeeze, translate, and do arithmetic with %nn fields.

BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and OUTREC now allow you to use a new JFY option to left-justify or right-justify the data in a field. For a left-justified field, leading blanks are removed and the characters from the first nonblank to the last nonblank are shifted left, with blanks inserted on the right if needed. For a right-justified field, trailing blanks are removed and the characters from the last nonblank to the first nonblank are shifted right, with blanks inserted on the left if needed.

Optionally for JFY, specific leading and trailing characters can be changed to blanks before justification begins, a leading string can be inserted, a trailing string can be inserted, and the output length can be changed.

BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and OUTREC now allow you to use a new SQZ option to left-squeeze or right-squeeze the data in a field. For a left-squeezed field, all blanks are removed and the characters from the first nonblank to the last nonblank are shifted left, with blanks inserted on the right if needed. For a right-squeezed field, all blanks are removed and the characters from the last nonblank to the first nonblank are shifted right, with blanks inserted on the left if needed.

Optionally for SQZ, specific characters can be changed to blanks before squeezing begins, a leading string can be inserted, a trailing string can be inserted, a string can be inserted wherever a group of blanks is removed between the first nonblank and the last nonblank, blanks can be kept as is between paired apostrophes or paired quotes, and the output length can be changed.

BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and OUTREC now allow you to insert past and future dates (relative to the date of your DFSORT run) into your records in various forms using new DATEn+r, DATEn-r, DATEn(c)+r, DATEn(c)-r, DATEnP+r and DATEnP-r constants. &DATEn+r, &DATEn-r, &DATEn(c)+r, &DATEn(c)-r, &DATEnP+r and &DATEnP-r can be used as aliases for DATEn+r, DATEn-r, DATEn(c)+r, DATEn(c)-r, DATEnP+r and DATEnP-r, respectively.

TRAILERx, BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and OUTREC now allow you to use FL format to convert 4-byte or 8-byte hexadecimal floating-point values to integer values.

TRAILERx, HEADERx, BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and OUTREC now allow you to use new TO=PDF and TO=PDC options to convert numeric values to PD values with F or C for the positive sign, respectively. The TO=PDC option is equivalent to the existing TO=PD option.

BUILD, OVERLAY, IFTHEN BUILD, IFTHEN OVERLAY and OUTREC now allow you to use &DATEn, &DATEn(c), &DATEnP, &YDDD=(abc), &YDDDNS=(ab), &TIMEn, &TIMEn(c) and &TIMEnP as aliases for DATEn, DATEn(c), DATEnP, YDDD=(abc), YDDDNS=(ab), TIMEn, TIMEn(c) and TIMEnP, respectively.

INCLUDE, OMIT and IFTHEN WHEN now allow you to compare date fields in various formats to past and future dates (relative to the date of your DFSORT run) using new DATEn+r, DATEn-r, DATEn(c)+r,

DATEn(c)-r, DATEnP+r, DATEnP-r, Y'DATEn'+r and Y'DATEn'-r constants. &DATEn+r, &DATEn-r, &DATEn(c)+r, &DATEn(c)-r, &DATEnP+r and &DATEnP-r can be used as aliases for DATEn+r, DATEn-r, DATEn(c)+r, DATEn(c)-r, DATEnP+r and DATEnP-r, respectively.

INCLUDE, OMIT and IFTHEN WHEN now allow you to test a field for numerics (field,EQ,NUM) or non-numerics (field,NE,NUM) in character (FS), zoned decimal (ZD) or packed decimal (PD) format.

INCLUDE, OMIT and IFTHEN WHEN now allow you to use &DATEn, &DATEn(c) and &DATEnP as aliases for DATEn, DATEn(c) and DATEnP, respectively.

Symbol Enhancements

A symbol can now be used for a %nn parsed field. For example, if Account,%01 is defined in SYMNAMES, Account can be used for %01. A symbol for %nn can be used in DFSORT control statements where %nn can be used. A symbol for %nn results in substitution of %nn.

A symbol can now be used for an output column. For example, if Start_address,18 is defined in SYMNAMES, Start_address: can be used for 18:. symbol: can be used in DFSORT control statements where c: can be used. A symbol for p or p,m or p,m,f results in substitution of p: for symbol: (output column).

A symbol can now be used for a new system symbol string constant. symbol,S'string' can be used to define a string containing any combination of EBCDIC characters and system symbols you want to use to form a character string. For example, if whererun,S'&JOBNAME. on &SYSPLEX' is defined in SYMNAMES, whererun can be used for the resulting constant. You can use dynamic system symbols such as &JOBNAME, &DAY, and so on, system-defined static system symbols such as &SYSNAME, &SYSPLEX, and so on, and installation-defined static system symbols specified by your installation in an IEASYMxx member of SYS1.PARMLIB.

A symbol for a system symbol string can be used in DFSORT and ICETOOL control statements where a symbol for a character string can be used. DFSORT will replace each system symbol in S'string' with its substitution text to create a character string in the format C'new_string'.

ICETOOL Enhancements

DISPLAY now allows you to use FL format to convert 4-byte or 8-byte hexadecimal floating-point values to integer values.

DISPLAY and OCCUR now allow you to use a new TBETWEEN(n) option to specify the number of blanks between title elements (title, page number, date, time).

SELECT and SPLICE now allow you to use an INREC statement to reformat your records before they are selected or spliced. All of the operands of the INREC statement (PARSE, BUILD, OVERLAY, IFTHEN, IFOUTLEN and FIELDS) are now available with SELECT and SPLICE.

SORT and MERGE Enhancements

The maximum length for a PD or ZD sort or merge field has been raised to 256.

SUM Enhancements

The maximum position for the end of a sum field has been raised to 32752.

Other Enhancements

DFSORT supports large physical sequential data sets for input, output and work data sets.

DSA can now be specified as a run-time option. This allows you to adjust the maximum amount of storage available to DFSORT for dynamic storage adjustment of individual Blockset sort applications when SIZE/MAINSIZE=MAX is in effect.

DFSORT now accepts and ignores zero values in the starting and ending address of the RECORD statement image in the 24-Bit Parameter List. You can set these addresses to zero if you don't want to pass a control statement to DFSORT using the third and fourth words of the parameter list

- END OF DOCUMENT -

---

by George Lewycky TIS-Financial Systems (646) 252-8882
george.lewycky@nyct.com

Thursday, April 20, 2006 DFSORT_feature.doc
Wednesday, May 24, 2006 DFSORT_feature_updated.doc
Wednesday, June 14, 2006 DFSORT_feature_updated.doc