User Guide for DFSORT PTFs UQ95214 and UQ95213

December, 2004

Frank L. Yaeger

DFSORT Team
IBM Systems Software Development
San Jose, California
Internet: yaeger@us.ibm.com

DFSORT/MVS Web Site

For papers, online books, news, tips, examples and more, visit the DFSORT home page at URL:

http://www.ibm.com/storage/dfsort

Abstract

This paper is the documentation for z/OS DFSORT V1R5 PTF **UQ95214** and DFSORT R14 PTF **UQ95213**, which were first made available in **December, 2004**. These PTFs provide important enhancements to DFSORT and DFSORT's ICETOOL for conditionally reformatting records; overlaying only selected parts of records; larger numeric fields and constants; new data formats for extracting digits, and displaying TOD and ETOD date and time values; restarting sequence numbers; join and match operations; date and time constants; conversion of statistical and count values; reports; easier migration from other sort products, and more. This paper highlights, describes, and shows examples of the new features provided by these PTFs for DFSORT and for DFSORT's powerful, multi-purpose ICETOOL utility. It also details new and changed messages associated with these PTFs.

Contents

User Guide for DFSORT PTFs UQ95214 and UQ95213

Introduction

DFSORT is IBM's high performance sort, merge, copy, analysis and reporting product. DFSORT is an optional feature of z/OS and OS/390.

DFSORT, together with DFSMS/MVS and RACF, form the strategic product base for the evolving system-managed storage environment. DFSMS/MVS provides vital storage and data management functions. RACF adds security functions. DFSORT adds the ability to do faster and easier sorting, merging, copying, reporting and analysis of your business information, as well as versatile data handling at the record, field and bit level.

DFSORT includes the versatile ICETOOL utility and the high-performance ICEGENER facility.

z/OS DFSORT V1R5 PTF **UQ95214** and DFSORT Release 14 PTF **UQ95213**, which were first made available in **December, 2004**, provide important enhancements to DFSORT and DFSORT's ICETOOL for conditionally reformatting records; overlaying only selected parts of records; larger numeric fields and constants; new data formats for extracting digits, and displaying TOD and ETOD date and time values; restarting sequence numbers; join and match operations; date and time constants; conversion of statistical and count values; reports; easier migration from other sort products, and more.

This paper highlights, describes, and shows examples of the new features provided by these PTFs for DFSORT and for DFSORT's powerful, multi-purpose ICETOOL utility. It also details new and changed messages associated with these PTFs.

You can access all of the DFSORT books online by clicking the **Publications** link on the DFSORT home page at URL:

http://www.ibm.com/storage/dfsort

This paper provides the documentation you need to start using the features and messages associated with z/OS DFSORT PTF UQ95214 or DFSORT R14 PTF UQ95213. The information in this paper will be included in the z/OS DFSORT books at their next major update (but not in the DFSORT R14 books).

You should refer to *z/OS DFSORT Application Programming Guide* for general information on DFSORT and ICETOOL features, and in particular for the framework of existing DFSORT features upon which these new features are built. You should refer to *z/OS DFSORT Messages, Codes and Diagnosis Guide* for general information on DFSORT messages.

Summary of Enhancements

ICETOOL Enhancements

ICETOOL's DISPLAY, OCCUR, RANGE, SELECT, SPLICE, STATS, UNIQUE and VERIFY operators now allow you to use larger numeric values for ON and BREAK fields. PD, ZD and FS fields can now be up to 31 digits (or more in some cases). BI and FI fields can now be up to 8 bytes (or more in some cases).

ICETOOL's RANGE operator now allows you to use larger decimal values for the HIGHER(n), LOWER(n), EQUAL(n) and NOTEQUAL(n) options. These values can now be up to 31 digits.

ICETOOL's DISPLAY, OCCUR, RANGE, SELECT, SPLICE, STATS and UNIQUE operators now allow you to use new UFF (unsigned free form) and SFF (signed free form) formats for ON and BREAK fields. UFF extracts a positive numeric value from a free form field (for example, ' $1234.56' is treated as +123456). SFF extracts a positive or negative value from a free form field (e.g., '(1,234.56)' is treated as -123456).

ICETOOL's DISPLAY and OCCUR operators now allow you to use new DC1-DC3 (TOD date), DE1-DE3 (ETOD date), TC1-TC4 (TOD time) and TE1-TE4 (ETOD time) formats for ON and BREAK fields. These new formats produce meaningful representations of TOD and ETOD date and time values.

ICETOOL's DISPLAY and OCCUR operators now allow you to specify multiline headings for the columns of your reports. You can specify one, two or three line headings with the HEADER option.

ICETOOL's DISPLAY and OCCUR operators now allow you to use new edit masks G1-G6 to display numeric values with 4 decimal digits in various ways.

ICETOOL's DISPLAY and OCCUR operators now allow you to use new YDDD(abc) and YDDDNS(abc) options to insert the year (yyyy) and day of the year (ddd) of your ICETOOL run into your titles in various forms.

ICETOOL's COUNT operator now allows you to use a new RC4 option to set RC=4 (instead of RC=12) or RC=0 based on the count of records in a data set.

ICETOOL's SPLICE operator now allows you to use a new KEEPBASE option to keep the base records as well as the spliced records.

ICETOOL's SPLICE operator now allows you to use new VLENMAX or VLENOVLY options to set the length of spliced records to the maximum of the base or overlay record length, or to the overlay record length.

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 (e.g., 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 (e.g., 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 (e.g., 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 (e.g., PAGE=(M11,LENGTH=3)).

INREC and OUTREC Enhancements

INREC and OUTREC now allow 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 FIELDS: The existing FIELDS parameter, or its new alias of BUILD, allows you to reformat each record by specifying all of its items one by one. BUILD or FIELDS gives you complete control over the items you want in your reformatted INREC or OUTREC 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.

INREC and OUTREC now allow 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.

INREC and OUTREC now allow 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.

INREC and OUTREC now allow 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.

INREC and OUTREC now allow you to use a set field in the CHANGE option.

INREC and OUTREC now allow you to restart the sequence number when the binary value of a specified field changes.

INREC and OUTREC now allow 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.

INREC and OUTREC now allow you to use new 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.

SORT and MERGE Enhancements

SORT and MERGE now allow you to use larger FS values for control fields. These values can now be up to 32 digits.

SORT and MERGE now allow you to use new UFF (unsigned free form) and SFF (signed free form) formats for control fields.

SORT and MERGE now allow you to use other characters besides '+' (plus) as a positive sign in control field values for CSL, CST, ASL and AST formats. A '-' (minus) sign is treated as negative and any other sign (e.g., blank) is treated as positive.

INCLUDE and OMIT Enhancements

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

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.

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.

SUM Enhancement

SUM now allows you to use larger ZD values for sum fields. These values can now be up to 31 digits.

Other Enhancements

For sort and copy applications, with concatenated variable-length input data sets for SORTIN, DFSORT now uses the largest LRECL it finds in the concatenation.

DFSORT now allows control statements in SYSIN and SORTCNTL to contain labels up to 70 characters, and allows any character in the label. DFSORT now ignores statements with a label followed only by blanks.

Operational Changes that may Require User Action

The following are operational changes that may require user action for existing DFSORT/ICETOOL applications that use certain functions as specified:

Return Area for ICETOOL STATS Operator

In order to handle larger numbers, ICETOOL will now return 16-byte PD values rather than 8-byte PD values in the return area for the STATS operator. If you have programs that call ICETOOL and use the returned STATS values, change the DCs for those values from PL8 to PL16 and recompile the programs before running them against this level of DFSORT/ICETOOL.

16-Byte FS Fields

In order to handle larger FS fields (a sign and up to 31 digits), DFSORT and ICETOOL will now treat a 16-byte FS fields as having a maximum of 16 digits rather than 15 digits.

For ICETOOL's DISPLAY and OCCUR operators, ON and BREAK fields for 16-byte FS values may result in different formatting for the output reports. You can use the new U15 formatting item to limit these values to 15 digits instead of 16 digits. For example, ON(11,16,FS,U15).

For INREC, OUTREC and OUTFIL OUTREC, 16-byte FS values may be edited or converted differently for output. You can use EDIT=(pattern) or LENGTH=n to change the length of the output fields, if appropriate. For example, 11,16,FS,EDIT=(SIIIIIIIIIIIIIT),SIGNS=(+,-).

TO=BI and TO=FI for 10-digit values

In order to handle larger numbers, INREC, OUTREC and OUTFIL OUTREC will create an 8-byte field instead of a 4-byte field when TO=BI or TO=FI is used for a 10-digit value. You can use LENGTH=n to change the length of the output fields, if appropriate. For example, 11,10,ZD,TO=BI,LENGTH=4.

New reserved words for Symbols

The following are new DFSORT/ICETOOL reserved words (uppercase only as shown) which are no longer allowed as symbols: DC1, DC2, DC3, DE1, DE2, DE3, SFF, TC1, TC2, TC3, TC4, TE1, TE2, TE3, TE4, UFF, ZDC and ZDF.

If you used these words as symbols, you must change them to other words, such as lowercase or mixed case forms of these words (for example, dc1 or Dc1).

Free form formats (SFF and UFF)

General Description

DFSORT and ICETOOL now allow you to use new SFF (signed free form numeric) and UFF (unsigned free form numeric) data formats to extract numeric values from free form fields consisting of decimal digits and any other characters (e.g., decimal point, separators, letters, blanks, parens, etc). This allows you to directly process values containing decimal digits and other characters (e.g., '123,456.78-') in the same way you would process ZD values.

You can now use the SFF and UFF formats in the SORT, MERGE, INCLUDE, OMIT, INREC, OUTREC and OUTFIL statements, and in the ON and BREAK parameters for all ICETOOL operators. See the individual Enhancement sections below for additional details.

Free Form Format Descriptions

- SFF (signed free form numeric)

  This format extracts decimal digits (0-9) from right to left anywhere in the field to form a positive or negative number. If - or) is found anywhere in the field, the number is treated as negative, otherwise it is treated as positive. Any combination of characters is valid, but characters other than 0-9, - and) are ignored.

  Example of extracted SFF values

```
Input value              Extracted value
--------------------     ---------------
     358,272,300.10         +35827230010
     358,272,300.1           +3582723001
        358,272,300           +358272300
         (82,316.90)            -8231690
          12-31-2004           -12312004
  G1***    52 $ 21 R              +15221
  G1***  ) 52 $ 21 R              -15221
000128637.240                  +128637240
400.52-                            -40052
   $400.5                           +4005
    173/821/9072/@3          +17382190723
   X,Y,Z                              +0
```

- UFF (unsigned free form numeric)

  This format extracts decimal digits (0-9) from right to left anywhere in the field to form a positive number. Any combination of characters is valid, but characters other than 0-9 are ignored.

  Example of extracted UFF values

```
Input value              Extracted value
--------------------     ---------------
      $58,272,300.1            +582723001
         $58,272,300            +58272300
          12-31-2004            +12312004
   (402)-125-3721XXX          +4021253721
  G1***    52 $ 21 R              +15221
000128637.240                 +128637240
+400.52                           +40052
+400.1                             +4001
    173/821/9072/@3          +17382190723
       ABC                             +0
```

Sample Syntax for DFSORT

```
* Keep records in which date values like this:
* yyyy-mm-dd
* yyyy/mm/dd
* are greater than 2004/11/12.
  INCLUDE COND=(21,10,UFF,GE,+20041121)

* Sort descending on the date values, and
* ascending on numeric values like this:
* (d,ddd,ddd.dd)
*  d,ddd,ddd.dd
  SORT FIELDS=(21,10,UFF,D,5,14,SFF,A)

* Total the numeric values and reformat the total.
  OUTFIL TRAILER1=('Total is ',TOT=(5,14,SFF,M5))
```

Sample Syntax for ICETOOL

```
* Select first duplicate records based on
* numeric values with digits and other characters.
  SELECT FROM(IN) TO(OUT) ON(128,40,SFF) FIRSTDUP
```

Larger fields and decimal constants

General Description

DFSORT and ICETOOL can now handle up to 31-digit signed values for ZD, PD and FS fields in all cases, removing the previous limit of 15-digit values wherever that limit was applicable. DFSORT and ICETOOL can also handle up to 31-digit signed values for SFF fields (new) and up to 31-digit unsigned values for UFF fields (new).

DFSORT and ICETOOL can now handle up to 8-byte signed values for FI fields, and up to 8-byte unsigned values for BI fields, removing the previous limit of 4-byte values wherever that limit was applicable.

You can now use these larger fields in the SORT, MERGE, INCLUDE, OMIT, SUM, INREC, OUTREC and OUTFIL statements, and in the ON and BREAK parameters for all ICETOOL operators. See the individual Enhancement sections below for additional details.

DFSORT can now handle larger decimal constants in the INCLUDE, OMIT, INREC, OUTREC and OUTFIL statements, and in ICETOOL's RANGE operator. See the individual Enhancement sections below for additional details.

Sample Syntax for DFSORT

```
SORT FIELDS=(2,26,FS,A)
OUTREC FIELDS=(31,18,ZD,M26,X,25:50,10,PD,M26,X,
  60:31,18,ZD,DIV,50,10,PD,TO=FS,LENGTH=19)
OUTFIL TRAILER1=(TOT=(60,19,FS,M26))
```

Sample Syntax for ICETOOL

```
DISPLAY FROM(IN) LIST(RPT) ON(28,6,FI) ON(1,23,FS)-
  BTITLE('Break value:') BREAK(39,31,ZD) -
  BTOTAL('Break total:')
STATS FROM(IN) ON(28,6,FI) ON(1,23,FS) ON(39,31,ZD)
```

TOD and ETOD formats (DCn, TCn, DEn, TEn)

General Description

DFSORT and ICETOOL now allow you to use new DCn (TOD date) and TCn (TOD time) data formats to produce meaningful representations of TOD (STCK) date and time values, and new DEn (ETOD date) and TEn (ETOD time) data formats to produce meaningful representations of ETOD (STCKE) date and time values.

DCn and DEn produce ZD values with variations of the year, month and day (e.g., Z'yyyymmdd') which can be further edited or converted into other forms (e.g., C'yyyy/mm/dd'). TCn and TEn produce ZD values with variations of the hours, minutes and seconds (e.g., Z'hhmmss') which can be further edited or converted into other forms (e.g., C'hh.mm.ss').

You can now use the DCn, TCn, DEn and TEn formats in the INREC, OUTREC and OUTFIL statements, and in the ON and BREAK parameters for ICETOOL operators DISPLAY and OCCUR. See the individual Enhancements sections below for additional details.

TOD and ETOD Format Descriptions

- DC1 (TOD date interpreted as Z'yyyymmdd')

  The 8 bytes of an input clock value, in the basic time-of day (TOD) format, is converted to a Z'yyyymmdd' value. The STCKCONV macro is used to do the conversion. yyyy represents the four-digit year, mm represents the month (00-12) and dd represents the day (00-31).

- DC2 (TOD date interpreted as Z'yyyymm')

  The 8 bytes of an input clock value, in the basic time-of day (TOD) format, is converted to a Z'yyyymm' value. The STCKCONV macro is used to do the conversion. yyyy represents the four-digit year and mm represents the month (00-12).

- DC3 (TOD date interpreted as Z'yyyyddd')

  The 8 bytes of an input clock value, in the basic time-of day (TOD) format, is converted to a Z'yyyyddd' value. The STCKCONV macro is used to do the conversion. yyyy represents the four-digit year and ddd represents the day of the year (000-366).

- DE1 (ETOD date interpreted as Z'yyyymmdd')

  The first 8 bytes of an input clock value, in the extended time-of day (ETOD) format, is converted to a Z'yyyymmdd' value. The STCKCONV macro is used to do the conversion. yyyy represents the four-digit year, mm represents the month (00-12) and dd represents the day (00-31).

- DE2 (ETOD date interpreted as Z'yyyymm')

  The first 8 bytes of an input clock value, in the extended time-of day (ETOD) format is converted to a Z'yyyymm' value. The STCKCONV macro is used to do the conversion. yyyy represents the four-digit year and mm represents the month (00-12).

- DE3 (ETOD date interpreted as Z'yyyyddd')

  The first 8 bytes of an input clock value, in the extended time-of day (ETOD) format is converted to a Z'yyyyddd' value. The STCKCONV macro is used to do the conversion. yyyy represents the four-digit year and ddd represents the day of the year (000-366).

- TC1 (TOD time interpreted as Z'hhmmss')

  The 8 bytes of an input clock value, in the basic time-of day (TOD) format, is converted to a Z'hhmmss' value. The STCKCONV macro is used to do the conversion. hh represents the hour (00-23), mm represents the minutes (00-59) and ss represents the seconds (00-59).

- TC2 (TOD time interpreted as Z'hhmm')

  The 8 bytes of an input clock value, in the basic time-of-day (TOD) format, is converted to a Z'hhmm' value. The STCKCONV macro is used to do the conversion. hh represents the hour (00-23) and mm represents the minutes (00-59).

- TC3 (TOD time interpreted as Z'hh')

  The 8 bytes of an input clock value, in the basic time-of-day (TOD) format, is converted to a Z'hh' value. The STCKCONV macro is used to do the conversion. hh represents the hour (00-23).

- TC4 (TOD time interpreted as Z'hhmmssxx')

  The 8 bytes of an input clock value, in the basic time-of day (TOD) format, is converted to a Z'hhmmssxx' value. The STCKCONV macro is used to do the conversion. hh represents the hour (00-23), mm represents the minutes (00-59), ss represents the seconds (00-59) and xx represents hundredths of a second (00-99).

- TE1 (ETOD time interpreted as Z'hhmmss')

  The first 8 bytes of an input clock value, in the extended time-of day (ETOD) format, is converted to a Z'hhmmss' value. The STCKCONV macro is used to do the conversion. hh represents the hour (00-23), mm represents the minutes (00-59) and ss represents the seconds (00-59).

- TE2 (ETOD time interpreted as Z'hhmm')

  The first 8 bytes of an input clock value, in the extended time-of day (ETOD) format, is converted to a Z'hhmm' value. The STCKCONV macro is used to do the conversion. hh represents the hour (00-23) and mm represents the minutes (00-59).

- TE3 (ETOD time interpreted as Z'hh')

  The first 8 bytes of an input clock value, in the extended time-of day (ETOD) format, is converted to a Z'hh' value. The STCKCONV macro is used to do the conversion. hh represents the hour (00-23).

- TE4 (ETOD time interpreted as Z'hhmmssxx')

  The first 8 bytes of an input clock value, in the extended time-of day (ETOD) format, is converted to a Z'hhmmssxx' value. The STCKCONV macro is used to do the conversion. hh represents the hour (00-23), mm represents the minutes (00-59), ss represents the seconds (00-59) and xx represents hundredths of a second (00-99).

Sample Syntax for DFSORT

```
* Display a TOD date as C'yyyy/mm/dd' and a
* TOD time as 'hh:mm:ss.xx'.
  OUTREC FIELDS=(1,10,X,26,8,DC1,EDIT=(TTTT/TT/TT),X,
    26,8,TC4,EDIT=(TT:TT:TT.TT),X,12,8)
```

Sample Syntax for ICETOOL

```
* Show the number of occurrences for each TOD date
* interpreted as Z'yyyyddd' and displayed as C'yyyy-ddd'.
  OCCUR FROM(IN) LIST(OUT) ON(15,8,DC3,E'9999-999) ON(VALCNT)
```

Year and day separators (YDDD and YDDDNS)

General Description

DFSORT and ICETOOL now allow you to display the current date with variations of the year (yyyy or yy) and day of the year (ddd). YDDD produces a date with a separator (e.g., 'ddd/yyyy'). YDDDNS produces a date without a separator (e.g., 'yyddd').

You can now use the YDDD and YDDDNS operands in the INREC, OUTREC and OUTFIL statements, and in ICETOOL operators DISPLAY and OCCUR. See the individual Enhancements sections below for additional details.

YDDD and YDDDNS Operand Descriptions

- YDDD=(abc) or YDDD(abc)

  YDDD=(abc) for DFSORT or YDDD(abc) for ICETOOL specifies that the current date is to appear in the form 'acb', where a and b indicate the order in which the year and day of the year are to appear and whether the year is to appear as two or four digits, and c is the character to be used to separate the year and day of the year.

  For a and b, use D to represent the day of the year (001-366), Y to represent the last two digits of the year (e.g., 04), or 4 to represent the four digits of the year (e.g., 2004). D, and Y or 4 can each be specified only once. Examples: YDDD=(DY-) would produce a date of the form 'ddd-yy' which on April 7, 2004, would appear as '098-04'. YDDD=(4D/) would produce a date of the form 'yyyy/ddd' which on April 7, 2004, would appear as '2004/098'.

  a, b and c must be specified.

- YDDDNS=(ab) or YDDDNS(ab)

  YDDDNS=(ab) for DFSORT or YDDDNS(ab) for ICETOOL specifies that the current date is to appear in the form 'ab', where a and b indicate the order in which the year and day of the year are to appear and whether the year is to appear as two or four digits.

  For a and b, use D to represent the day of the year (001-366), Y to represent the last two digits of the year (e.g., 04), or 4 to represent the four digits of the year (e.g., 2004). D, and Y or 4 can each be specified only once. Examples: YDDDNS=(DY) would produce a date of the form 'dddyy' which on April 7, 2004, would appear as '09804'. YDDDNS=(4D) would produce a date of the form 'yyyyddd' which on April 7, 2004, would appear as '2004098'.

  a and b must be specified.

Sample Syntax for DFSORT

```
* Display the current date as C'ddd/yyyy'.
  OUTFIL HEADER2=(5:'Printed on ',YDDD=(D4/),30:'Page',PAGE)
```

Sample Syntax for ICETOOL

```
* Display the current date as C'yyddd'.
DISPLAY FROM(IN) LIST(RPT) TITLE('Printed on') YDDDNS(YD) PAGE -
  HEADER('Volume') ON(11,6,CH)
```

Reformatting Enhancements (INREC, OUTREC, OUTFIL)

BUILD, OVERLAY and IFTHEN

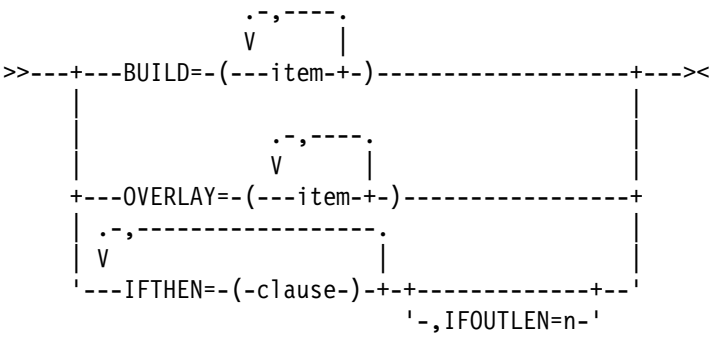
General Description

DFSORT's INREC, OUTREC and OUTFIL statements now provide much more powerful capabilities to reformat your records. Two important new operands, OVERLAY and IFTHEN, are now available in addition to the existing FIELDS operand for INREC and OUTREC, and the existing OUTREC operand for OUTFIL.

For consistency, BUILD can now be used in INREC and OUTREC as an alias for FIELDS, and in OUTFIL as an alias for OUTREC. Of course, you can still use the FIELDS operand for INREC and OUTREC, and the OUTREC operand for OUTFIL, but you might want to start using BUILD so you don't have to remember which operand goes with which statement. BUILD will be used for consistency in this document rather than FIELDS or OUTREC.

You can use all of the new and existing reformatting items with BUILD, OVERLAY and IFTHEN (e.g., fields, constants, arithmetic expressions, edit masks, conversion, lookup and change, sequence numbers, etc).

BUILD, OVERLAY and IFTHEN Operands Syntax



BUILD, OVERLAY and IFTHEN Operand Descriptions

You can create reformatted records in one of the following three ways using unedited, edited, or converted input fields and a variety of constants:

- **BUILD:** Reformat each record by specifying all of its item one by one. Build gives you complete control over the items you want in your reformatted records and the order in which they appear. You can delete, rearrange and insert fields and constants.

  Example:

  ```
  INREC BUILD=(1,20,C'ABC',26:5C'*',
    15,3,PD,EDIT=(TTT.TT),21,30,80:X)
  ```

  Note: BUILD is an alias for and equivalent to the existing FIELDS operand of INREC and OUTREC and to the existing OUTREC operand of OUTFIL.

- **OVERLAY:** Reformat each record by specifying just the items that overlay specific columns. Overlay lets you change specific existing columns without affecting the entire record.

  Example:

  ```
  OUTREC OVERLAY=(45:45,8,TRAN=LTOU)
  ```

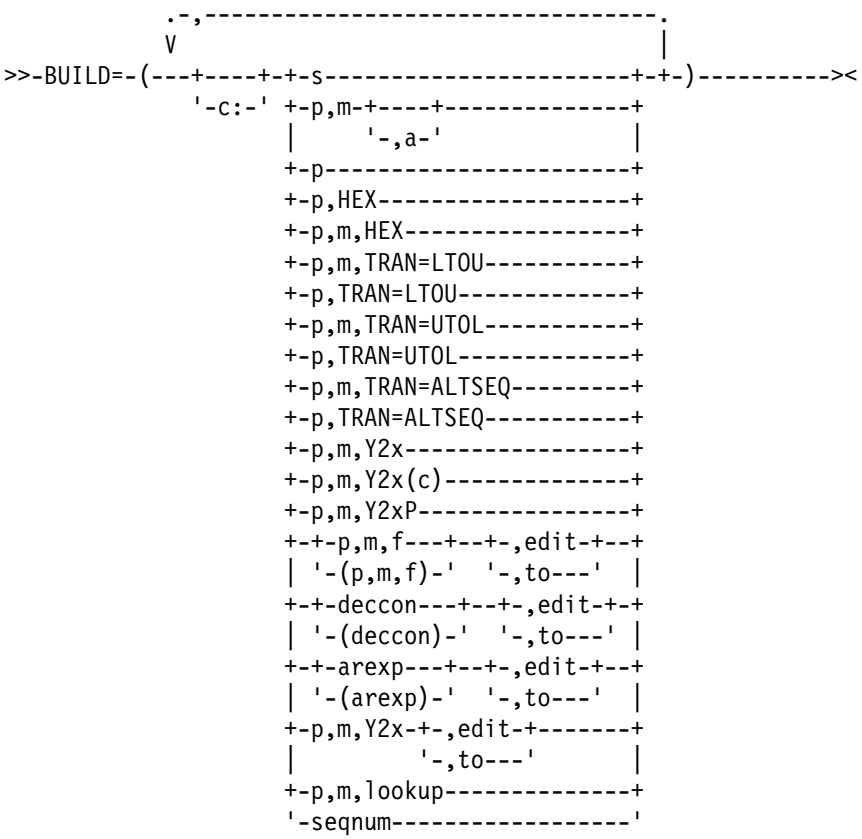
- **IFTHEN clauses:** Reformat different records in different ways by specifying how build or overlay items are applied to records that meet given criteria. IFTHEN clauses let you use sophisticated conditional logic to choose how different record types are reformatted.

  Example:

  ```
  OUTFIL IFTHEN=(WHEN=(1,5,CH,EQ,C'TYPE1'),
           BUILD=(1,40,C'**',+1,TO=PD)),
         IFTHEN=(WHEN=(1,5,CH,EQ,C'TYPE2'),
           BUILD=(1,40,+2,TO=PD,X'FFFF')),
         IFTHEN=(WHEN=NONE,OVERLAY=(45:C'NONE'))
  ```

Together, the BUILD, OVERLAY and IFTHEN operands for INREC, OUTREC and OUTFIL give you a new level of flexibility for reformatting your records at various stages of DFSORT processing.

BUILD Operand Syntax



BUILD Operand Description

See the descriptions of the OUTREC operand of the OUTFIL statement, the FIELDS operand of the INREC statement, and the FIELDS operand of the OUTREC statement, in *z/OS DFSORT Application Programming Guide* for complete details of the BUILD operand and all of the reformatting items you can use with it.

OVERLAY Operand Syntax

OVERLAY Operand Description

Specifies each item that is to overlay specific columns in the reformatted record. Columns that are not overlaid remain unchanged. If you want to insert, rearrange, or delete fields, use BUILD rather than OVERLAY. Use OVERLAY only to overlay existing columns or to add fields at the end of every record. OVERLAY can be easier to use then BUILD when you just want to change a few fields without rebuilding the entire record.

For fixed-length records, the first input and output data byte starts at position 1. For variable-length records, the first input and output data byte starts at position 5, after the RDW in positions 1-4.

Use c: (column) to specify the output positions to be overlaid. If you do not specify c: for the first item, it defaults to 1:. If you do not specify c: for any other item, it starts after the previous item. For example, if you specify:

```
INREC OVERLAY=(25,2,11:C'A',15,3,C'**')
```

Input positions 25-26 are placed at output positions 1-2; C'A' is placed at output position 11; input positions 15-17 are placed at output positions 12-14; and C'**' is placed at output positions 15-16. The rest of the record remains unchanged.

You can specify items in any order, you can change the same item multiple times and you can overlap output columns. **Changes to earlier items affect changes to later items.** For example, say you specify:

```
OUTFIL OVERLAY=(21:8,4,ZD,MUL,+10,TO=ZD,LENGTH=6,
    5:5,1,TRAN=UTOL,
    5:5,1,CHANGE=(1,C'a',C'X',C'b',C'Y'),NOMATCH=(5,1))
```

and input position 5 has 'A'. The second item (UTOL) would change 'A' to 'a' and the third item (CHANGE) would change 'a' again to 'X'.

If you specify an OVERLAY item that extends the overlay record beyond the end of the input record, the reformatted record length is automatically increased to that length, and blanks are filled in on the left as needed. For

variable-length records, the RDW length is increased to correspond to the larger reformatted record length after all of the OVERLAY items are processed. For example, if your input record has a length of 40 and you specify:

```
OUTREC OVERLAY=(16:C'ABC',51:5C'*',35:15,2)
```

the output record is given a length of 55. Blanks are filled in from columns 41-50. For variable-length records, the length in the RDW is changed from 40 to 55 after all of the OVERLAY items are processed.

Missing bytes in specified input fields are replaced with blanks so the padded fields can be processed.

For OUTFIL statements:

- You can use the OVERLAY parameter with the FTOV parameter to convert fixed-length record data sets to variable-length record data sets.
- You can use the VLTRIM parameter with the OVERLAY parameter to remove specified trailing bytes from the end of variable-length records.
- You can use the OVERLAY parameter with any or all of the report parameters in the same way as for the BUILD parameter.
- The OVERLAY parameter of the OUTREC statement applies to all input records whereas the OVERLAY parameter of the OUTFIL statement only applies to the OUTFIL input records for its OUTFIL group.

See the descriptions of the OUTREC operand of the OUTFIL statement, the FIELDS operand of the INREC statement, and the FIELDS operand of the OUTREC statement, in *z/OS DFSORT Application Programming Guide* for details of the items listed in the OVERLAY syntax diagram above.

You can specify all of the items for OVERLAY in the same way that you can specify them for BUILD with the following exceptions:

- You cannot specify p or p,HEX or p,TRAN=value for OVERLAY. Note that p can be specified without m as the last item in BUILD for an INREC, OUTREC or OUTFIL statement, even if another one of these statements has OVERLAY. For example:

  ```
  INREC FIELDS=(1,18,8C'*',23)
  OUTREC OVERLAY=(24:C'A')
  ```

- You cannot specify / for OVERLAY (/ is only allowed in BUILD of IFTHEN BUILD for OUTFIL).
- For p,m,H or p,m,F or p,m,D fields specified for OVERLAY, fields are aligned as necessary without changing the preceding bytes.
- For variable-length records, you must not overlay positions 1-4 (the RDW) for OVERLAY, so be sure to specify the first column (c:) as 5 or greater. Do not specify 1:, 2:, 3: or 4: anywhere in your OVERLAY parameter. If you do not specify the first column, it will default to 1: which is invalid for variable-length records with OVERLAY. Whereas BUILD=(1,m,...) is required, OVERLAY=(1,m) is not allowed since it would overlay the RDW.

Sample Syntax:

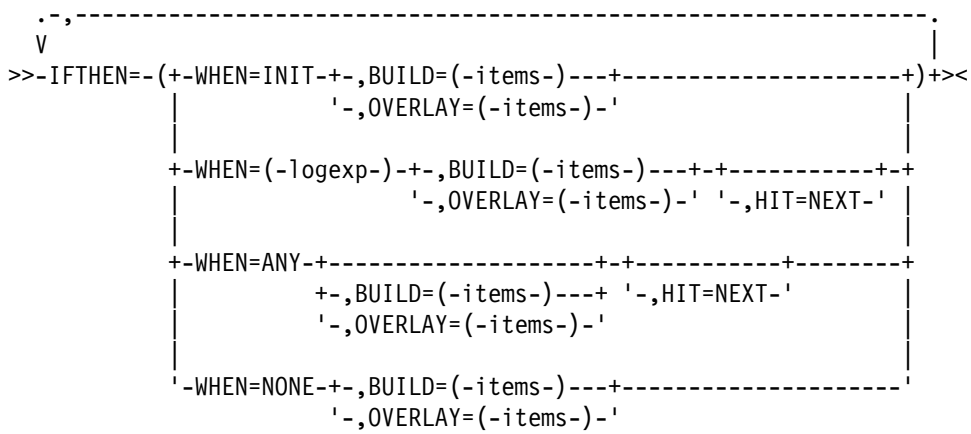
Fixed input records:

```
OUTFIL FNAMES=FLR,
  OVERLAY=(21:21,4,ZD,TO=PD,LENGTH=4,
     2:5,8,HEX,45:C'*',32,4,C'*',81:SEQNUM,5,ZD)
```

Variable input records:

```
  OUTFIL FNAMES=VLR,
    OVERLAY=(5:X'0001',28:C'Date is ',YDDDNS=(4D),
         17:5C'*')
```

IFTHEN Operand Syntax



IFTHEN Operand Description

IFTHEN clauses allow you to reformat different records in different ways by specifying how build or overlay items are to be applied to records that meet given criteria. IFTHEN clauses let you use simple or complex conditional logic to choose how different record types are reformatted.

If you want to insert, rearrange, or delete fields in the same way for every record, use BUILD rather than IFTHEN. If you want to overlay existing columns in the same way for every record, use OVERLAY rather than IFTHEN. Use IFTHEN clauses if you want to insert, rearrange, delete or overlay fields in different ways for different records.

You can use four types of IFTHEN clauses as follows:

- **WHEN=INIT:** Use one or more WHEN=INIT clauses to apply build or overlay items to all of your input records. WHEN=INIT clauses are processed before any of the other IFTHEN clauses.
- **WHEN=(logexp):** Use one or more WHEN=(logexp) clauses to apply build or overlay items to your input records that meet specified criteria. A WHEN=(logexp) clause is satisfied when the logical expression evaluates as true.
- **WHEN=ANY:** Use a WHEN=ANY clause after multiple WHEN=(logexp) clauses to apply additional build or overlay items to your input records if they satisfied the criteria for any of the preceding WHEN=(logexp) clauses.
- **WHEN=NONE:** Use one or more WHEN=NONE clauses to apply build or overlay items to your input records that did not meet the criteria for any of the WHEN=(logexp) clauses. WHEN=NONE clauses are processed after any of the other IFTHEN clauses. If you do not specify a WHEN=NONE clause, only the WHEN=INIT changes (if any) are applied to input records that do not meet the criteria for any of the WHEN=(logexp) clauses.

IFTHEN clauses are processed in the following order:

- WHEN=INIT clauses
- WHEN=(logexp) clauses and WHEN=ANY clauses
- WHEN=NONE clauses

Processing of IFTHEN clauses continues unless one of the following occurs:

- A WHEN=(logexp) or WHEN=ANY clause is satisfied, and HIT=NEXT is not specified.
- For an OUTFIL statement, a WHEN=(logexp), WHEN=ANY or WHEN=NONE clause is satisfied, and BUILD with / is specified (multiple output records). Note that / is only allowed in BUILD for an OUTFIL statement, not for an INREC or OUTREC statement.
- There are no more IFTHEN clauses to process. When processing of IFTHEN clauses stops, the IFTHEN record created so far is used as the output record.

Example:

```
OUTFIL IFTHEN=(WHEN=(12,1,BI,ALL,X'3F'),OVERLAY=(18:C'Yes')),
       IFTHEN=(WHEN=(35,2,PD,EQ,+14),BUILD=(1,40,45,3,HEX),HIT=NEXT),
       IFTHEN=(WHEN=(35,2,PD,GT,+17),BUILD=(1,40,41,5,HEX),HIT=NEXT),
       IFTHEN=(WHEN=ANY,BUILD=(1,55,C'ABC',70:X)),
       IFTHEN=(WHEN=(35,2,PD,EQ,+8),BUILD=(1,40,2/,45,10)),
       IFTHEN=(WHEN=(63,2,CH,EQ,C'AB'),OVERLAY=(18:C'No')),
       IFTHEN=(WHEN=NONE,BUILD=(1,40,51,8,TRAN=LTOU))
```

For this example, the IFTHEN clauses are processed as follows:

- If IFTHEN clause 1 is satisfied, its overlay item is applied and IFTHEN processing stops.
- If IFTHEN clause 1 is not satisifed, its overlay item is not applied and IFTHEN processing continues.
- If IFTHEN clause 2 is satisfied, its build items are applied and IFTHEN processing continues.
- If IFTHEN clause 2 is not satisfied, its build items are not applied and IFTHEN processing continues.
- If IFTHEN clause 3 is satisfied, its build items are applied and IFTHEN processing continues.
- If IFTHEN clause 3 is not satisfied, its build items are not applied and IFTHEN processing continues.
- If IFTHEN clause 4 is satisfied, its build items are applied and IFTHEN processing stops.
- If IFTHEN clause 4 is not satisifed, its build items are not applied and IFTHEN processing continues.
- If IFTHEN clause 5 is satisfied, its build items are applied and IFTHEN processing stops.
- If IFTHEN clause 5 is not satisifed, its build items are not applied and IFTHEN processing continues.
- If IFTHEN clause 6 is satisfied, its overlay item is applied and IFTHEN processing stops.
- If IFTHEN clause 6 is not satisifed, its overlay item is not applied and IFTHEN processing continues.
- If IFTHEN clause 7 is satisfied, its build items are applied and IFTHEN processing stops.
- If IFTHEN clause 7 is not satisifed, its build items are not applied and IFTHEN processing stops.

All of the IFTHEN clauses operate sequentially on an IFTHEN record. The IFTHEN record is created initially from the input record. Each IFTHEN clause tests and changes the IFTHEN record, as appropriate. Thus, **changes made by earlier IFTHEN clauses are "seen" by later IFTHEN clauses**. For example, if you have a 40-byte input record and specify:

```
OUTREC IFTHEN=(WHEN=INIT,OVERLAY=(8:8,4,ZD,ADD,+1,TO=ZD,LENGTH=4)),
       IFTHEN=(WHEN=(8,4,ZD,EQ,+27),OVERLAY=(28:C'Yes')),
       IFTHEN=(WHEN=NONE,OVERLAY=(28:C'No'))
```

The WHEN=INIT clause adds 1 to the ZD value and stores it in the IFTHEN record. The WHEN=(8,4,ZD,EQ,+27) clause tests the incremented ZD value in the IFTHEN record rather than the original ZD value in the input record.

The IFTHEN record is adjusted as needed for the records created or changed by the IFTHEN clauses. For fixed-length records, blanks are filled in on the left as needed. For variable-length records, the RDW length is adjusted as needed each time the IFTHEN record is changed.

Missing bytes in specified input fields are replaced with blanks so the padded fields can be processed.

DFSORT sets an appropriate LRECL (or maximum record length) for the output records based on the build and overlay items specified by the IFTHEN clauses. However, DFSORT does not analyze the possible results of WHEN=(logexp) conditions when determining an appropriate LRECL. When you use IFTHEN clauses, you can override the LRECL determined by DFSORT with the IFOUTLEN parameter.

If SEQNUM is used in multiple IFTHEN clauses, the sequence number will be incremented for each record that has the SEQNUM item applied to it. For example, if your input is:

```
RECORD A 1
RECORD B 1
RECORD B 2
RECORD C 1
RECORD A 2
RECORD C 2
RECORD B 3
```

and you specify:

```
  INREC IFTHEN=(WHEN=(8,1,CH,EQ,C'A'),OVERLAY=(15:SEQNUM,4,ZD)),
    IFTHEN=(WHEN=(8,1,CH,EQ,C'B'),OVERLAY=(15:SEQNUM,4,ZD)),
    IFTHEN=(WHEN=NONE,OVERLAY=(15:SEQNUM,4,ZD))
```

your output will be:

```
RECORD A 1    0001
RECORD B 1    0001
RECORD B 2    0002
RECORD C 1    0001
RECORD A 2    0002
RECORD C 2    0002
RECORD B 3    0003
```

For OUTFIL statements:

- You can use IFTHEN clauses with the FTOV parameter to convert fixed-length record data sets to variable-length record data sets.
- You can use the VLTRIM parameter with IFTHEN clauses to remove specified trailing bytes from the end of variable-length records.
- You can use IFTHEN clauses with any or all of the report parameters in the same way as for the BUILD parameter.
- The IFTHEN clauses of the OUTREC statement apply to all input records whereas the IFTHEN clauses of the OUTFIL statement only apply to the OUTFIL input records for its OUTFIL group.

BUILD in an INREC, OUTREC or OUTFIL statement can differ from IFTHEN in another one of these statements with respect to p without m as the last item. For example:

```
  INREC BUILD=(1,24,32:25)
  OUTREC IFTHEN=(WHEN=(8,1,ZD,GT,+5),BUILD=(1,24,25:C'Yes',28,10)),
         IFTHEN=(WHEN=NONE,BUILD=(1,24,25:C'No ',28,10))
```

WHEN=INIT Clause Description

A WHEN=INIT clause specifies build or overlay items to be applied to all records. If multiple WHEN=INIT clauses are specified, they are processed in the order in which they are specified. WHEN=INIT clauses are processed before any other type of IFTHEN clauses.

WHEN=INIT

Identifies a WHEN=INIT clause.

BUILD=(items)

Specifies the build items to be applied to each record. See "BUILD Operand Description" for details. You can specify all of the items in the same way that you can specify them for INREC, OUTREC or OUTFIL BUILD. / to create blank records or new records is not allowed with WHEN=INIT.

OVERLAY=(items)

Specifies the overlay items to be applied to each record. See "OVERLAY Operand Description" for details. You can specify all of the items in the same way that you can specify them for INREC, OUTREC or OUTFIL OVERLAY.

Sample Syntax:

```
INREC IFTHEN=(WHEN=INIT,BUILD=(1,20,21:C'Department',31:3X,21,60)),
      IFTHEN=(WHEN=(5,2,CH,EQ,C'D1'),OVERLAY=(31:8,3)),
      IFTHEN=(WHEN=(5,2,CH,EQ,C'D2'),OVERLAY=(31:12,3))
```

WHEN=(logexp) Clause Description

A WHEN=(logexp) clause specifies build or overlay items to be applied to records for which the specified logical expression is true. If multiple WHEN=(logexp) clauses are specified, they are processed in the order in which they are specified.

WHEN=(logexp)

Identifies a WHEN=(logexp) clause and specifies the criteria to be tested to determine if the build or overlay items are to be applied to the record. See the description of the INCLUDE statement in *z/OS DFSORT Application Programming Guide* for details of the logical expressions you can use. You can specify all of the logical expressions in the same way that you can specify them for the INCLUDE statement except that:

- You cannot specify FORMAT=f with WHEN=(logexp).
- You cannot specify D2 format in WHEN=(logexp).
- Locale processing is not used for WHEN=(logexp).
- VLSCMP and VLSHRT are not used with WHEN=(logexp). Instead, missing bytes in specified input fields are replaced with blanks so the padded fields can be processed.

BUILD=(items)

Specifies the build items to be applied to each record for which the logical expression is true. See "BUILD Operand Description" for details. You can specify all of the items in the same way that you can specify them for INREC, OUTREC or OUTFIL BUILD. Note that although / can be used to create blank records and new records with OUTFIL, it cannot be used with INREC or OUTREC.

OVERLAY=(items)

Specifies the overlay items to be applied to each record for which the logical expression is true. See "OVERLAY Operand Description" for details. You can specify all of the items in the same way that you can specify them for INREC, OUTREC or OUTFIL OVERLAY.

HIT=NEXT

Specifies that IFTHEN processing should continue even if the logical expression is true. By default (if HIT=NEXT is not specified), IFTHEN processing stops if the logical expression is true.

Sample Syntax:

```
OUTFIL FNAMES=OUT2,
   IFTHEN=(WHEN=(1,3,CH,EQ,C'T01',AND,
        18,4,ZD,LE,+2000),OVERLAY=(42:C'Type1 <= 2000'),HIT=NEXT),
   IFTHEN=(WHEN=(1,3,CH,EQ,C'T01',AND,6,1,BI,BO,X'03'),
      BUILD=(1,21,42,13)),
   IFTHEN=(WHEN=(1,3,CH,EQ,C'T01',AND,
        18,4,ZD,GT,+2000),OVERLAY=(42:C'Type1 > 2000 '),HIT=NEXT),
   IFTHEN=(WHEN=(1,3,CH,EQ,C'T01',AND,6,1,BI,BO,X'01'),
      BUILD=(1,25,42,13))
```

WHEN=ANY Clause Description

A WHEN=ANY clause specifies build or overlay items to be applied to records for which the logical expression in any "preceding" WHEN=(logexp) clause is true. For the first WHEN=ANY clause, the "preceding" WHEN=(logexp) clauses are those before this WHEN=ANY clause. For the second or subsequent WHEN=ANY clause, the "preceding" WHEN=(logexp) clauses are those between the previous WHEN=ANY clause and this WHEN=ANY clause. At least one WHEN=(logexp) clause must be specified before a WHEN=ANY clause. A WHEN=ANY clause can be used without any build or overlay items to just stop IFTHEN processing if any preceding WHEN=(logexp) clause is satisfied.

BUILD=(items)

Specifies the build items to be applied to each record for which the logical expression in any preceding WHEN=(logexp) clause is true. See "BUILD Operand Description" for details. You can specify all of the items in the same way that you can specify them for INREC, OUTREC or OUTFIL BUILD. Note that although / can be used to create blank records and new records with OUTFIL, it cannot be used with INREC or OUTREC.

OVERLAY=(items)

Specifies the overlay items to be applied to each record for which the logical expression in any prceding WHEN=(logexp) clause is true. See "OVERLAY Operand Description" for details. You can specify all of the items in the same way that you can specify them for INREC, OUTREC or OUTFIL OVERLAY.

HIT=NEXT

Specifies that IFTHEN processing should continue even if the logical expression in a preceding WHEN=(logexp) clause is true. By default (if HIT=NEXT is not specified), IFTHEN processing stops if the logical expression in a preceding WHEN=(logexp) clause is true.

Sample Syntax:

```
OUTREC IFTHEN=(WHEN=(1,3,SS,EQ,C'T01,T02,T03'),
           BUILD=(C'Group A',X,1,80),HIT=NEXT),
 IFTHEN=(WHEN=(1,3,SS,EQ,C'T04,T05,T06'),
           BUILD=(C'Group B',X,1,80),HIT=NEXT),
 IFTHEN=(WHEN=(1,3,SS,EQ,C'T07,T08,T09,T10'),
           BUILD=(C'Group C',X,1,80),HIT=NEXT),
 IFTHEN=(WHEN=ANY,OVERLAY=(16:C'Group Found'))
```

WHEN=NONE Clause Description

A WHEN=NONE clause specifies build or overlay items to be applied to records for which none of the logical expressions in any WHEN=(logexp) clause is true. If there are no WHEN=(logexp) clauses, the build or overlay

items are applied to all of the records. If multiple WHEN=NONE clauses are specified, they are processed in the order in which they are specified. WHEN=NONE clauses are processed after any other type of IFTHEN clauses.

BUILD=(items)

Specifies the build items to be applied to each record for which no logical expression was true. See "BUILD Operand Description" for details. You can specify all of the items in the same way that you can specify them for INREC, OUTREC or OUTFIL BUILD. Note that although / can be used to create blank records and new records with OUTFIL, it cannot be used with INREC or OUTREC.

OVERLAY=(items)

Specifies the overlay items to be applied to each record for which no logical expression was true. See "OVERLAY Operand Description" for details. You can specify all of the items in the same way that you can specify them for INREC, OUTREC or OUTFIL OVERLAY.

Sample Syntax:

```
OUTFIL FNAMES=OUT4,
  IFTHEN=(WHEN=INIT,BUILD=(1,20,21:C'Department',31:3X,21,60)),
  IFTHEN=(WHEN=(5,2,CH,EQ,C'D1'),OVERLAY=(31:8,3)),
  IFTHEN=(WHEN=(5,2,CH,EQ,C'D2'),OVERLAY=(31:12,3)),
  IFTHEN=(WHEN=NONE,OVERLAY=(31:C'***'))
```

IFOUTLEN Operand Syntax

```
>>--IFOUTLEN=n--><
```

IFOUTLEN Operand Description

Overrides the LRECL (or maximum record length) determined by DFSORT from your IFTHEN clauses. DFSORT sets an appropriate LRECL for the output records based on the build and overlay items specified by the IFTHEN clauses. However, DFSORT does not analyze the possible results of WHEN=(logexp) conditions when determining an appropriate LRECL. When you use IFTHEN clauses, you can override the LRECL determined by DFSORT with the IFOUTLEN parameter.

Fixed-length records longer than the IFOUTLEN length are truncated to the IFOUTLEN length. Fixed-length records shorter than the IFOUTLEN are padded with blanks to the IFOUTLEN length. Variable-length records longer than the IFOUTLEN length are truncated to the IFOUTLEN length.

n specifies the length to use for the LRECL (or maximum record length). The value for n must be between 1 and 32767, but must not be larger than the maximum LRECL allowed for the RECFM, and must not conflict with the specified or retrieved LRECL for the fixed-length output data set.

Sample Syntax:

```
OUTREC IFOUTLEN=70,
  IFTHEN=(WHEN=(5,1,CH,EQ,C'1',AND,8,3,ZD,EQ,+10),
      BUILD=(1,40,C'T01-GROUP-A',65)),
  IFTHEN=(WHEN=(5,1,CH,EQ,C'2',AND,8,3,ZD,EQ,+12),
      BUILD=(1,40,C'T02-GROUP-B',65))
```

RESTART=(p,m) for seqnum

General Description

DFSORT's BUILD, OVERLAY, IFTHEN BUILD and IFTHEN OVERLAY parameters of the INREC, OUTREC and OUTFIL OUTREC statements now give you the capability to restart the sequence number at the start value each time a specified input field changes. You can use the new RESTART=(p,m) subparameter with the other seqnum reformatting item subparameters when you want to restart the sequence number.

seqnum Syntax and Description

```
>>-SEQNUM,n,fs-+-----------------+-+----------------+-->
               '-,START=-+-j---+-' '-,INCR=-+-i---+-'
                         '-(j)-'            '-(i)-'

>-+----------------+-><
   '-,RESTART=(p,m)-'
```

See the description of the "seqnum" parameter of the INREC, OUTREC and OUTFIL statements in *z/OS DFSORT Application Programming Guide* for details of n, fs, START, j, INCR and i.

RESTART Subparameter Description

RESTART=(p,m) specifies that DFSORT is to restart the sequence number at the starting value (START=j) when the binary value for the specified input field (p,m) changes. This allows you to sequence each set of records with the same value (that is, duplicate records) separately. For example: Without RESTART, if you had six input records with 'A', 'A', 'A', 'B', 'B' and 'C', respectively, in position 1, the output records would be given the sequence numbers 1, 2, 3, 4, 5 and 6. But with RESTART=(1,1), the output records are given the sequence numbers 1, 2, 3, 1, 2 and 1; DFSORT restarts at the starting value (1, by default) when the input value at position 1 changes from 'A' to 'B' and again when it changes from 'B' to 'C'.

p,m specifies the input field to be used to determine when the sequence number is to be restarted at the starting value.

If a variable-length input record is too short to contain a specified restart field, zeros will be used for the missing bytes, intentionally or unintentionally.

p specifies the first byte of the input field relative to the beginning of the input record. The first data byte of a fixed-length record has relative position 1. The first data byte of a variable-length record has relative position 5, because the first four bytes are occupied by the RDW. The field cannot extend beyond byte 32752.

m specifies the length in bytes of the input field. The value for m must be between 1 and 256.

Sample Syntax:

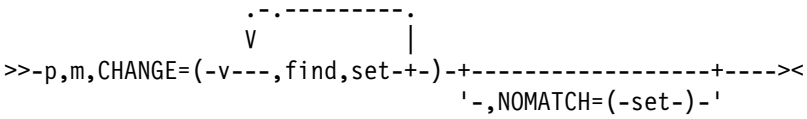
```
* Restart the sequence number at 1000 whenever the
* value in positions 35-42 changes.
  INREC OVERLAY=(81:SEQNUM,8,ZD,START=1000,RESTART=(35,8))
```

p,m for CHANGE

General Description

DFSORT's BUILD, OVERLAY, IFTHEN BUILD and IFTHEN OVERLAY parameters of the INREC, OUTREC and OUTFIL OUTREC statements now give you the capability to select an input field for a match with lookup and change as well as for a non-match. The CHANGE operand now allows an input field (p,m), as well as a constant, for the set parameter in the same way that NOMATCH does.

lookup and change Parameter Syntax and Description



See the description of the "p,m,lookup" parameter of the INREC, OUTREC and OUTFIL statements in *z/OS DFSORT Application Programming Guide* for details of p, m, CHANGE, v, find and NOMATCH.

set for CHANGE Subparameter Description

set specifies a set constant or set field to be used as the output field if the corresponding find constant matches the input field value. Set constants and set fields can be intermixed.

Set constants can be either character string constants (C'xx...x') or hexadecimal string constants (X'yy...yy') of 1 to v bytes. See the description of the INCLUDE statement in *z/OS DFSORT Application Programming Guide* for details of coding character and hexadecimal string constants.

If the string is less than v bytes, it will be padded on the right to a length of v bytes, using blanks (X'40') for a character string constant or zeros (X'00') for a hexadecimal string constant.

Set fields are specified as q,n.

q specifies the first byte of the input field relative to the beginning of the input record. The first data byte of a fixed-length record has relative position 1. The first data byte of a variable-length record has relative position 5, because the first four bytes are occupied by the RDW. The field cannot extend beyond byte 32752.

n specifies the input length of 1 to v bytes. If n is less than v, the input field will be padded on the right to a length of v bytes, using blanks (X'40').

Sample Syntax:

```
INREC OVERLAY=(11,1,
        CHANGE=(6,
           C'R',51,6,
           C'U',C'UPDATE',
           X'FF',C'EMPTY',
           C'X',35,6,
           C'A',C'ALTER')
        NOMATCH=(21,6))
```

SFF and UFF formats

DFSORT's new SFF and UFF formats, described earlier in "Free Form Formats (SFF and UFF)", can be used in the following reformatting items:

- p,m,f,edit (edited field)
- p,m,f,to (converted field)
- arexp,edit (edited arithmetic expression)
- arexp,to (converted arithmetic expression)

Note: See the discussion of these reformatting items in *z/OS DFSORT Application Programming Guide* for details on using these items. SFF and UFF can be used for input fields in the same way as the other formats.

You can use 1-44 bytes for SFF and UFF fields. A maximum of 31 digits is allowed. If a UFF or SFF value with more than 31 digits is found, the leftmost digits are ignored. If p,m,SFF or p,m,UFF is specified without edit or to, the M0 edit mask is used by default.

See the next section for some additional information on using SFF and UFF fields in reformatting items.

Sample Syntax

```
* Convert C'sddd,ddd.ddd' values to Z'ddddddddd' values.
  INREC OVERLAY=(21:21,12,SFF,TO=ZD,LENGTH=9)
  SORT FIELDS=(5,8,CH,A)
* SUM the Z'ddddddddd' values.
  SUM FIELDS=(21,9,ZD)
* Convert the Z'ddddddddd' values back to
* C'sddd,ddd.ddd' values.
  OUTREC OVERLAY=(21:21,9,ZD,
    EDIT=(SIII,IIT.TTT),SIGNS=(+,-))
```

Larger constants and ZD, PD, FS, BI and FI fields

DFSORT can now handle larger fields and constants in the following reformatting items:

- p,m,f,edit (edited field)
- p,m,f,to (converted field)
- deccon,edit (editied decimal constant)
- deccon,to (converted decimal constant)
- arexp,edit (edited arithmetic expression)
- arexp,to (converted arithmetic expression)

Note: See the discussion of these reformatting items in *z/OS DFSORT Application Programming Guide* for the details to which the **changes** discussed in this section apply.

The limits have been changed to allow:

- up to 31 digits for decimal constants (the old limit was up to 15 digits)
- up to 31 digits for ZD, PD and FS input fields (e.g., p,m,ZD) and converted fields (e.g., TO=PD)
- up to 8 bytes (20 digits) for BI input fields (p,m,BI) and converted fields (TO=BI)
- up to 8 bytes (19 digits) for FI input fields (p,m,FI) and converted fields (TO=FI)

The old and new lengths for input fields and converted fields are as follows:

Item	Old length	New length
ZD field	1-15	1-31
PD field	1-8	1-16
FS field	1-16	1-32
BI field	1-4	1-8
FI field	1-4	1-8

The lengths for the new SFF and UFF input fields are 1-44 bytes.

Additionally:

- edit masks (e.g., M5) can now accomodate up to 31 digits.
- EDIT patterns can now accomodate up 44 characters with up to 31 digits.
- LENGTH=n can now have a value from 1-44 for n.

The digits needed (d) for editing or conversion are as follows:

Format	Input length	Digits needed (d)
ZD	m	m
PD	m	2m-1
FS	32	31
FS	m (<32)	m
SFF, UFF	32-44	31
SFF, UFF	m (<32)	m
BI, FI	1	3
BI, FI	2	5
BI, FI	3	8
BI, FI	4	10
BI, FI	5	13
BI, FI	6	15
BI, FI	7	17
BI, FI	8	20

Converted field lengths are as follows:

TO format	Converted length
ZD	d
PD	d/2 + 1
FS	d + 1
BI with d>=9	4
BI with d>9	8
FI with d>=9	4
FI with d>9	8

For BI conversion:

- An input value greater than 18446744073709551615 (X'FFFFFFFFFFFFFFFF') produces an output value of 18446744073709551615 (X'FFFFFFFFFFFFFFFF').
- An input value less than zero produces an absolute output value. For example, an input value of P'-5000' produces a BI output value of 5000 (X'1388').

For FI conversion, an input value greater than 9223372036854775807 (X'7FFFFFFFFFFFFFFF') produces an output value of 9223372036854775807 (X'7FFFFFFFFFFFFFFF'), and an input value less than -9223372036854775808 (X'8000000000000000') produces an output value of -9223372036854775808 (X'8000000000000000').

For decimal constants with 1 to 15 significant digits, the default (d) number of digits for editing or conversion is 15. For decimal constants with 16 to 31 significant digits, the default (d) number of digits for editing or conversion is 31. If EDIT is specified, the number of digits in the pattern (I's and T's) is used.

For arithmetic expressions:

- An arithmetic expression produces a signed, 31-digit zoned decimal (ZD) result to be edited or converted as specified.

- If an intermediate or final result of an arithmetic expression overflows 31 digits, the overflowing intermediate or final result will be truncated to 31 digits.
- The default number of digits (d) used for editing or conversion is 15 if **every** term in the expression is one of the following:
  - a 1-4 byte BI or FI field
  - a 1-8 byte PD field
  - a 1-15 byte ZD, FS, SFF or UFF field
  - a decimal constant with 1-15 significant digits

  The default number of digits (d) used for editing or conversion is 31 if **any** term in the expression is one of the following:
  - a 5-8 byte BI or FI field
  - a 9-16 byte PD field
  - a 16-31 byte ZD field
  - a 16-32 byte FS field
  - a 16-44 byte SFF or UFF field
  - a decimal constant with 16-31 significant digits

  If EDIT is specified, the number of digits in the pattern (I's and T's) is used.

Sample Syntax

```
OUTREC FIELDS=(31,18,ZD,M26,X,25:50,10,PD,M26,X,
  60:31,18,ZD,DIV,50,10,PD,TO=FS,LENGTH=19,X,
  -10000000000000000000,TO=PD)
```

DCn, TCn, DEn and TEn formats

DFSORT's new DC1-3, TC1-4, DE1-3 and TE1-4 formats, described earlier in "TOD and ETOD formats (DCn, TCn, DEn, TEn)", can be used in the p,m,f,edit (edited field) and p,m,f,to (converted field) reformatting items to produce meaningful representations of TOD (STCK) and ETOD (STCKE) date and time values.

Note: See the discussion of these reformatting items in *z/OS DFSORT Application Programming Guide* for details on using these items. DC1-3, TC1-4, DE1-3 and TE1-4 can be used for input fields in the same way as the other formats.

You must use 8 bytes for DCn, TCn, DEn and TEn fields. All values are treated as positive. If p,m,DCn or p,m,TCn or p,m,DEn or p,m,TEn is specified without edit or to, the M11 edit mask is used by default. The number of digits (d) used for editing or conversion is equal to the number of digits in the converted ZD value for the specific format (e.g., d=7 for DE3 which converts an ETOD date to Z'yyyyddd').

Sample Syntax

```
* For record type 1, display a TOD date as C'yyyy/ddd'
* and a TOD time as 'hh:mm:ss'.
* For record type 2, display a TOD date as C'yyyy-ddd'
* and a TOD time as 'hh.mm.ss'.
  INREC IFTHEN=(WHEN=(6,1,BI,EQ,1),
          BUILD=(1,10,X,26,8,DC3,EDIT=(TTTT/TTT),X,
            26,8,TC1,EDIT=(TT:TT:TT),X,12,8)),
        IFTHEN=(WHEN=(6,1,BI,EQ,2),
          BUILD=(1,10,X,26,8,DC3,EDIT=(TTTT-TTT),X,
            26,8,TC1,EDIT=(TT.TT.TT),X,12,8))
```

DATE, DATE=(abcd) and DATENS=(abc)

DFSORT now allows you to use DATE, DATE=(abcd) and DATENS=(abc) constants to insert the current date in your records in the same way you can insert them in report headers and trailers.

See the descriptions of DATE, DATE=(abcd) and DATENS=(abc) for the HEADER1 operand in *z/OS DFSORT Application Programming Guide* for complete details on these parameters.

Sample Syntax

```
* Add C'mm-dd-yyyy' to each record.
  OUTREC OVERLAY=(81:DATE=(MD4-))
```

YDDD=(abc) and YDDDNS=(ab)

DFSORT's new YDDD=(abc) and YDDDNS=(ab) parameters, described earlier in "Year and day separators (YDDD and YDDDNS)", give you another way to insert the current date in your records.

Sample Syntax

```
* Add C'ddd/yyyy' to each record.
  OUTFIL OVERLAY=(81:YDDD=(D4/))
```

TIME, TIME=(abc) and TIMENS=(ab)

DFSORT now allows you to use TIME, TIME=(abc) and TIMENS=(ab) constants to insert the current time in your records in the same way you can insert them in report headers and trailers.

See the descriptions of TIME, TIME=(abc) and TIMENS=(ab) for the HEADER1 operand in *z/OS DFSORT Application Programming Guide* for complete details on these parameters.

Sample Syntax

```
* Add C'hh:mm:ss' to each record.
  INREC OVERLAY=(81:TIME)
```

TO=ZDC and TO=ZDF

DFSORT now allows you to use two new conversion formats, ZDC and ZDF, in the p,m,f,to (converted field) and arexp,to (converted arithmetic expression) reformatting items (and anywhere else to can be used) to choose whether ZD output fields use C (ZDC) or F (ZDF or ZD) for a positive sign. TO=ZDF is equivalent to the old TO=ZD parameter and results in an F sign for positive values. TO=ZDC results in a C sign for positive values. A D sign is always used for negative values.

Sample Syntax

```
  OUTREC BUILD=(1,10,
* Use a C sign for the ZD results of an
* arithmetic expression.
    25,5,PD,DIV,+1024,TO=ZDC,LENGTH=9,
    41,40)
```

OUTFIL Trailer Enhancements

count=(to) and stats=(to)

DFSORT now allows you to use the following to convert counts and statistics in trailers to BI, FI, PD, ZD, ZDF, ZDC or FS format:

- COUNT=(to)
- TOTAL=(p,m,f,to)
- MIN=(p,m,f,to)
- MAX=(p,m,f,to)
- AVG=(p,m,f,to)
- SUBCOUNT=(to)
- SUBTOTAL=(p,m,f,to)
- SUBMIN=(p,m,f,to)
- SUBMAX=(p,m,f,to)
- SUBAVG=(p,m,f,to)

See the descriptions of the specific count and statistics parameters in *z/OS DFSORT Application Programming Guide* for details on using these parameters in trailers.

See the description of p,m,f,to in *z/OS DFSORT Application Programming Guide* for details of the to fields you can use.

Sample Syntax

```
  OUTFIL REMOVECC,NODETAIL,
* Get a count as a 4-byte PD value and
* a Total as a 5-byte PD value.
     TRAILER1=(COUNT=(TO=PD,LENGTH=4),
        TOTAL=(16,5,PD,TO=PD,LENGTH=5))
```

COUNT+n=(edit/to) and COUNT-n=(edit/to)

General Description

DFSORT now allows you to add 1-999 to a count in a trailer, or subtract 1-999 from a count in a trailer, and edit or convert the result.

COUNT+n Operand Description

COUNT+n=(edit) or COUNT+n=(to)

n is added to the 15-digit count before it is edited or converted. n can be 1 to 3 decimal digits. For example, if there are 6810 input records, COUNT+2=(M11,LENGTH=6) produces a count of '006812'. One important use for this parameter is to add 1 for the TRAILER1 record to the count of data records.

See the description of p,m,f,edit and p,m,f,to in *z/OS DFSORT Application Programming Guide* for details of the edit and to fields you can use.

Sample Syntax

```
OUTFIL TRAILER1=('Count of report records is ',
   COUNT+1=(TO=FS,LENGTH=7))
```

COUNT-n Operand Description

COUNT-n=(edit) or COUNT-n=(to)

n is subtracted from the 15-digit count before it is edited or converted. n can be 1 to 3 decimal digits. For example, if there are 6810 input records, COUNT-1=(M11,LENGTH=6) produces a count of '006809'. One important use for this parameter is to subtract 1 for a header record from the count of data records.

See the description of p,m,f,edit and p,m,f,to in *z/OS DFSORT Application Programming Guide* for details of the edit and to fields you can use.

Sample Syntax

```
OUTFIL TRAILER1=('Count of data records is ',
  COUNT-2=(M11,LENGTH=5))
```

PAGE=(edit/to)

DFSORT now allows you to use PAGE=(edit) to edit page numbers in trailers, or PAGE=(to) to convert page numbers in trailers. The 15-digit page number is edited or converted as specified. See the description of p,m,f,edit and p,m,f,to in *z/OS DFSORT Application Programming Guide* for details of the edit and to fields you can use.

Sample Syntax

```
OUTFIL TRAILER2=('Min is ',MIN=(28,5,PD,TO=ZDF),X,
                 'Max is ',MAX=(28,5,PD,TO=ZDF),//,
                 'End of page ',PAGE=(EDIT=(TTT)))
```

nX'string'

DFSORT now allows you to use nX'string' to insert hexadecimal constants in your trailers in the same way you can insert them in your reformatted records.

See the description of nX'string' in *z/OS DFSORT Application Programming Guide* for complete details.

Sample Syntax

```
* Create output record with:
* INPUT('count')
  OUTFIL REMOVECC,NODETAIL,
    TRAILER1=('INPUT(',
      X'7D',COUNT=(M11,LENGTH=5),X'7D',C')')
```

SFF and UFF formats

DFSORT's new SFF and UFF formats, described earlier in "Free Form Formats (SFF and UFF)", can be used in the following statistics in trailers:

- TOTAL=(p,m,f,edit/to)
- MIN=(p,m,f,edit/to)
- MAX=(p,m,f,edit/to)
- AVG=(p,m,f,edit/to)
- SUBTOTAL=(p,m,f,edit/to)
- SUBMIN=(p,m,f,edit/to)
- SUBMAX=(p,m,f,edit/to)
- SUBAVG=(p,m,f,edit/to)

See the descriptions of the specific statistics parameters in *z/OS DFSORT Application Programming Guide* for details on using these parameters in trailers.

See the description of p,m,f,edit and p,m,f,to in *z/OS DFSORT Application Programming Guide* for details of the edit and to fields you can use.

You can use 1-44 bytes for SFF and UFF fields. A maximum of 31 digits is allowed. If a UFF or SFF value with more than 31 digits is found, the leftmost digits are ignored.

See the next section for some additional information on using SFF and UFF fields in statistics.

Sample Syntax

```
* Get the department average for numeric values
* like this:
* dddd.dds
* and reformat the average.
  OUTFIL SECTIONS=(29,5,
   TRAILER3=('Average for department ',29,5,' is ',
     AVG=(5,14,SFF,EDIT=(IIIT.TTS),SIGNS=(,,+,-))))
```

Larger ZD, PD, FS, BI and FI statistics

DFSORT can now handle larger fields in the following statistics in trailers:

- TOTAL=(p,m,f,edit/to)
- MIN=(p,m,f,edit/to)
- MAX=(p,m,f,edit/to)
- AVG=(p,m,f,edit/to)

- SUBTOTAL=(p,m,f,edit/to)
- SUBMIN=(p,m,f,edit/to)
- SUBMAX=(p,m,f,edit/to)
- SUBAVG=(p,m,f,edit/to)

Note: See the discussion of these statistics in *z/OS DFSORT Application Programming Guide* for the details to which the **changes** discussed in this section apply. See "Larger constants and ZD, PD, FS, BI and FI fields" earlier in this document for details of the new limits for input fields, edit fields, and to fields.

For TOTAL and SUBTOTAL, the digits needed (d) for editing or conversion are as follows:

Format	Input length	Digits needed (d)
ZD	1-15	15
ZD	16-31	31
PD	1-8	15
PD	9-16	31
FS	1-15	15
FS	16-32	31
SFF, UFF	1-15	15
SFF, UFF	16-44	31
BI, FI	1-4	10
BI, FI	5-8	20

If EDIT is specified, the number of digits in the pattern (I's and T's) is used.

If a TOTAL or SUBTOTAL overflows 31 digits, the overflowing value will be truncated to 31 digits.

Sample Syntax

```
  OUTFIL TRAILER1=(2/,
   'Total:   ',TOTAL=(1031,17,ZD,
      EDIT=(TTI,TTT,TTT,TTT.TTTTTT)),/,
   'Average: ',AVG=(1031,17,ZD,
      EDIT=(TTT,TTT,TTT,TTT.TTTTTT)))
```

YDDD=(abc) and YDDDNS=(ab)

DFSORT's new YDDD=(abc) and YDDDNS=(ab) parameters, described earlier in "Year and day separators (YDDD and YDDDNS)", give you another way to insert the current date in your trailers.

Sample Syntax

```
* Display C'yy.ddd' in page trailer.
  OUTFIL TRAILER2=('Created on ',YDDD=(YD.))
```

OUTFIL Header Enhancements

PAGE=(edit/to)

DFSORT now allows you to use PAGE=(edit) to edit page numbers in headers, or PAGE=(to) to convert page numbers in headers. The 15-digit page number is edited or converted as specified. See the description of p,m,f,edit and p,m,f,to in *z/OS DFSORT Application Programming Guide* for details of the edit and to fields you can use.

Sample Syntax

```
OUTFIL HEADER2=(20:'- ',PAGE=(EDIT=(IT)),C' -')
```

nX'string'

DFSORT now allows you to use nX'string' to insert hexadecimal constants in your headers in the same way you can insert them in your reformatted records.

See the description of nX'string' in *z/OS DFSORT Application Programming Guide* for complete details.

Sample Syntax

```
OUTFIL HEADER1=(5X'FF',1,4,X'0302AB3E',5,4,5X'FF')
```

YDDD=(abc) and YDDDNS=(ab)

DFSORT's new YDDD=(abc) and YDDDNS=(ab) parameters, described earlier in "Year and day separators (YDDD and YDDDNS)", give you another way to insert the current date in your headers.

Sample Syntax

```
* Display C'dddyyyy' in page header.
  OUTFIL HEADER2=('Created on ',YDDDNS=(D4),2/)
```

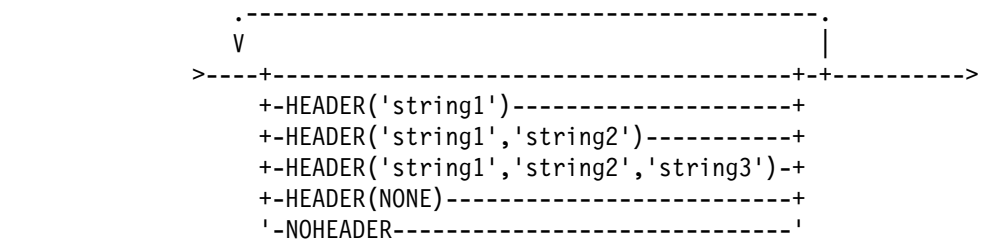
ICETOOL Enhancements

Multiline HEADER (DISPLAY, OCCUR)

General Description

ICETOOL's DISPLAY and OCCUR operators now allow you to specify multiline headings for the columns of your reports. You can specify one, two or three line headings with the HEADER operand.

HEADER Operand Syntax



See the description of HEADER(NONE) and NOHEADER in *z/OS DFSORT Application Programming Guide* for details of those operands.

HEADER('string1') Description

Specifies a heading to be printed for the corresponding ON field. The specified string is used instead of the standard column heading for the corresponding ON field. (ON fields and HEADER operands correspond one-for-one according to the order in which they are specified; that is, the first HEADER operand corresponds to the first ON field, the second HEADER operand corresponds to the second ON field, and so on.)

The string (1 to 50 characters) must be enclosed in single apostrophes. To include a single apostrophe (') in the string, specify two single apostrophes ('). If the string length is greater than the column width for the corresponding ON field, the column width is increased to the string length.

The heading is left-justified for character fields or right-justified for numeric fields and is underlined with hyphens for the entire column width (overriding the default of left-justified, non-underlined headings). Character values are left-justified and numeric values are right-justified (overriding the default of left-justified field values).

A null string ('') or blank string (' ') may be used to set string1 to blanks.

Blanks at the start or end of a heading string may alter the justification of the heading or the width of the column.

If HEADER('string1') is used for any ON field, HEADER('string1'), HEADER('string1','string2'), HEADER('string1','string2','string3') or HEADER(NONE) must be used for each ON field.

Sample Syntax

```
* Display headers like this:
* line1    line1
* ------   ------
  DISPLAY FROM(IN) LIST(RPT) BLANK -
    HEADER('line1') ON(5,6,CH) -
    HEADER('line2') ON(11,6,CH)
```

HEADER('string1','string2') Description

Specifies a heading to be printed for the corresponding ON field. A two-line heading is used with string1 on line1 and string2 on line2. A null string ('') or blank string (' ') may be used to set string1 or string2 to blanks. A comma (,) may also be used to set string1 to blanks. For example, HEADER(,'string1') results in blanks for this heading in line1 and string2 for this heading in line2.

If HEADER('string1','string2') is used for any ON field, HEADER('string1'), HEADER('string1','string2'), HEADER('string1','string2','string3') or HEADER(NONE) must be used for each ON field.

If a HEADER('string1','string2') operand is specified, a two-line heading is also used for any HEADER('string1') operands you specify, with string1 for that heading on line1 and blanks for that heading on line2. HEADER(,'string1') can be used to put blanks for that heading on line1 and string1 for that heading on line2.

See HEADER('string1') for more details on the HEADER operand.

Sample Syntax

```
* Display headers like this:
* line1
* line2    line2
* ------   ------
  DISPLAY FROM(IN) LIST(RPT) BLANK -
    HEADER('line1','line2') ON(5,6,CH) -
    HEADER(,'line2') ON(11,6,CH)
```

HEADER('string1','string2','string3') Description

Specifies a heading to be printed for the corresponding ON field. A three-line heading is used with string1 on line1, string2 on line2 and string3 on line3. A null string ('') or blank string (' ') may be used to set string1, string2 or string3 to blanks. A comma (,) may also be used to set string1 or string2 to blanks. For example, HEADER(,,'string3') results in blanks for this heading in line1 and line2 and string3 for this heading in line3.

If HEADER('string1','string2','string3') is used for any ON field, HEADER('string1'), HEADER('string1','string2'), HEADER('string1','string2','string3') or HEADER(NONE) must be used for each ON field.

If a HEADER('string1','string2','string3') operand is specified:

- a three-line heading is also used for any HEADER('string1') operands you specify, with string1 for that heading on line1 and blanks for that heading on line2 and line3. HEADER(,,'string1') can be used to put blanks for that heading on line1 and line2 and string1 for that heading on line3.
- a three-line heading is also used for any HEADER('string1','string2') operands you specify, with string1 for that heading on line1, string2 for that heading on line2 and blanks for that heading on line3. HEADER(,'string1','string2') can be used to put blanks for that heading on line1, string1 for that heading on line2 and string2 for that heading on line3.

See HEADER('string1') for more details on the HEADER operand.

Sample Syntax

```
* Display headers like this:
* line1
* line2    line2
* line3    line3    line3
*------   ------   ------
  DISPLAY FROM(IN) LIST(RPT) BLANK -
    HEADER('line1','line2','line3') ON(5,6,CH) -
    HEADER(,'line2','line3') ON(11,6,CH) -
    HEADER(,,'line3') ON(17,6,CH)
```

G1-G6 (DISPLAY, OCCUR)

ICETOOL's DISPLAY and OCCUR operators now allow you to specify new formatting items G1-G6 for use with:

- ON(p,m,f,formatting)
- ON(VLEN,formatting)
- ON(NUM,formatting)
- ON(VALCNT,formatting)
- BREAK(p,m,f,formatting)

The new G1-G6 items are similar to the old D1-D6 items, except that where D1-D6 provide 3 decimal places, G1-G6 provide 4 decimal places.

The table below describes the available Gn masks and shows how the values 12345678 and -1234567 would be printed for each mask. In the pattern:

- d is used to represent a decimal digit (0-9)
- w is used to represent a leading sign that will be blank for a positive value or - for a negative value
- x is used to represent a trailing sign that will be blank for a positive value or - for a negative value

```
Gn    Pattern                                    12345678   -1234567
-- ----------------------------------------  -----------  ---------
G1 wddd,ddd,ddd,ddd,ddd,ddd,ddd,ddd,ddd.dddd   1,234.5678  -123.4567
G2 wddd.ddd.ddd.ddd.ddd.ddd.ddd.ddd,dddd   1.234,5678  -123,4567
G3 wddd ddd ddd ddd ddd ddd ddd ddd ddd,dddd   1 234,5678  -123,4567
G4 wddd'ddd'ddd'ddd'ddd'ddd'ddd'ddd.dddd   1'234.5678  -123.4567
G5 wddd'ddd'ddd'ddd'ddd'ddd'ddd'ddd,dddd   1'234,5678  -123,4567
G6 ddd ddd ddd ddd ddd ddd ddd ddd ddd,ddddx  1 234,5678   123,4567-
```

Sample Syntax

```
* Display a report like this:
*  Four decimal
*        places             Count
* -------------   ---------------
* sddd,ddd.dddd                 d
  OCCUR FROM(IN) LIST(OUT1)-
   HEADER('Four','decimal','places') -
   ON(32,10,ZD,G1) -
   HEADER(,,'Count') ON(VALCNT)
```

Udd (DISPLAY, OCCUR)

ICETOOL's DISPLAY and OCCUR operators now allow you to specify new formatting item Udd for use with:

- ON(p,m,f,formatting)
- ON(VLEN,formatting)
- ON(NUM,formatting)
- ON(VALCNT,formatting)
- BREAK(p,m,f,formatting)

The new Udd item is similar to the old Ndd item. Ndd or Udd can be used to specify the number of digits for a numeric field in order to change the column width, or to prevent overflow for totals. dd specifies the number of digits and must be a two-digit number between 01 and 31 for ON(p,m,f,formatting), ON(VLEN,formatting) and BREAK(p,m,f,formatting), or between 01 and 15 for ON(NUM,formatting) and ON(VALCNT,formatting).

The default number of digits (d) for a numeric field is the maximum number of digits for that field. Udd can be used to increase or decrease d. (Ndd can only be used to increase d.) For example, if you use ON(11,12,ZD), d is 12. If you know that your ZD field cannot be more than 9 digits, you can use ON(11,12,ZD,U09) to decrease d from 12 to 9, and thus redcuce the width of the column.

The default number of digits (d) with TOTAL or BTOTAL is 15 or 31 depending on the number of digits for the numeric field. Udd can be used to increase or decrease d. (Ndd can only be used to increase d.) For example, if

you use ON(11,9,ZD) with TOTAL, d is 15. If you know that your TOTAL cannot be more than 10 digits, you can use ON(11,9,ZD,U10) to decrease d from 15 to 10, and thus reduce the width of the column.

If you use Udd and a numeric value or total overflows dd digits, ICETOOL prints asterisks for that numeric value or total and terminates the operation. You can prevent the overflow by specifying an appropriately higher dd value for Udd. For example, if ON(11,12,ZD,U09) results in overflow, you can use ON(1,12,ZD,U10) instead.

Sample Syntax

```
* Use Unn to reduce the column width to the
* maximum size for the TOTAL.
  DISPLAY FROM(IN) LIST(OUT2) PLUS -
   HEADER('ZD9') ON(11,9,ZD,U10) TOTAL('Total')
```

KEEPBASE (SPLICE)

General Description

ICETOOL's SPLICE operator now allows you to use a new KEEPBASE option to keep the base records as well as the spliced records.

KEEPBASE can be used to keep the base records (first duplicate) as well as the spliced records. The base records will be unchanged.

To illustrate the splicing process when KEEPBASE is specified, if we had the following six records with the base fields, ON fields and WITH fields as shown:

```
UNIQA  ONA
BASEA  ONB
DUPAA  ONB    WITHA
UNIQB  ONC
BASEB  OND
DUPBB  OND    WITHB
```

The two base records with duplicates (first ONB record and first OND record) would be kept along with the two spliced records (ONB and OND). The resulting four unspliced and spliced output records would be:

```
BASEA  ONB
BASEA  ONB    WITHA
BASEB  OND
BASEB  OND    WITHB
```

Note that without KEEPBASE, the two base records with duplicates (first ONB record and first OND record) would not be kept. The resulting two spliced output records would be:

```
BASEA  ONB    WITHA
BASEB  OND    WITHB
```

If we used KEEPNODUPS and KEEPBASE with the original six records, the resulting six unspliced and spliced output records would be:

```
UNIQA  ONA
BASEA  ONB
BASEA  ONB    WITHA
UNIQB  ONC
BASEB  OND
BASEB  OND    WITHB
```

KEEPBASE Option Description

KEEPBASE specifies that base records (first duplicate) are to be kept as well as spliced records. The base records will be unchanged.

Sample Syntax

```
SPLICE FROM(COMBINED) TO(OUT) -
  ON(1,12,CH) WITHALL KEEPBASE KEEPNODUPS -
  WITH(24,10) WITH(53,13) WITH(66,1)
```

VLENMAX and VLENOVLY (SPLICE)

General Description

ICETOOL's SPLICE operator now allows you to use new VLENMAX or VLENOVLY options to set the length of spliced records to the maximum of the base or overlay record length, or to the overlay record length.

For variable-length records, by default (without VLENMAX or VLENOVLY), the spliced record has the same length as the base record, and WITH fields from the overlay record that are beyond the end of the base record do not appear in the spliced record. For example, with WITHALL, if we had the following four records with the lengths (in the RDW), ON field and WITH fields as shown:

```
30 | BASE1 ON1           BASE2
25 |       ON1     WITHA
50 |       ON1     WITHB         WITHC       WITHD
40 |       ON1     WITHE         WITHF
```

the resulting three spliced output records would be:

```
30 | BASE1 ON1     WITHA BASE2
30 | BASE1 ON1     WITHB BASE2
30 | BASE1 ON1     WITHE BASE2
```

The WITHC, WITHD and WITHF fields are beyond the end of the base record, so they are not spliced.

However, if you specify **VLENMAX**, the spliced record is given the larger of the base record length or overlay record length. If you specify **VLENOVLY**, the spliced record is given the overlay record length. In either case, if the overlay record length is larger, bytes in the extended spliced record that are not overlaid are filled in with blanks.

The resulting three spliced output records with WITHALL and VLENMAX would be:

```
30 | BASE1 ON1     WITHA BASE2
50 | BASE1 ON1     WITHB BASE2   WITHC       WITHD
40 | BASE1 ON1     WITHE BASE2   WITHF
```

The resulting three spliced output records with WITHALL and VLENOVLY would be:

```
25 | BASE1 ON1     WITHA
50 | BASE1 ON1     WITHB BASE2   WITHC       WITHD
40 | BASE1 ON1     WITHE BASE2   WITHF
```

VLENMAX Option Description

VLENMAX specifies that for variable-length records, the length of the spliced record is set to the maximum length of the base record and overlay record. VLENMAX overrides the default of setting the length of the spliced record to the length of the base record.

If VLENMAX is specified with or without WITHALL, the spliced record is given the larger of the base record length or overlay record length. If the overlay record length is larger than the base record length, bytes in the extended spliced record that are not overlaid are filled in with blanks.

If VLENMAX is specified with WITHEACH, the spliced record is given the largest of the base record length or overlay record lengths. If the largest overlay record length is larger than the base record length, bytes in the extended spliced record that are not overlaid are filled in with blanks.

For fixed-length records, VLENMAX is ignored since the base, overlay and spliced records all have the same length.

Sample Syntax

```
SPLICE FROM(CON) TO(OUT) ON(5,5,CH) WITHALL -
   WITH(12,5) WITH(22,20) VLENMAX
```

VLENOVLY Option Description

VLENOVLY specifies that for variable-length records, the length of the spliced record is set to the length of the overlay record. VLENOVLY overrides the default of setting the length of the spliced record to the length of the base record.

If VLENOVLY is specified with or without WITHALL, the spliced record is given the overlay record length. If the overlay record length is larger than the base record length, bytes in the extended spliced record that are not overlaid are filled in with blanks. If the overlay record length is smaller than the base record length, bytes at the end of the base record do not appear in the spliced record.

VLENOVLY cannot be specified with WITHEACH.

For fixed-length records, VLENOVLY is ignored since the base, overlay and spliced records all have the same length.

Sample Syntax

```
SPLICE FROM(CON) TO(OUT) ON(5,5,CH) WITHALL -
   WITH(12,5) WITH(22,20) VLENOVLY
```

RC4 (COUNT)

General Description

ICETOOL's COUNT operator now allows you to use a new RC4 option to set RC=4 (instead of RC=12) or RC=0 based on the count of records in a data set.

If EMPTY, NOTEMPTY, HIGHER(x), LOWER(y), EQUAL(v) or NOTEQUAL(w) is specified, ICETOOL checks the record count as determined by its E35 user exit against the specified criteria. If the criteria is met (e.g., HIGHER(20) is specified and the record count is 21 or more), ICETOOL sets RC=12 for the COUNT operator by default or RC=4 for the COUNT operator if RC4 is specified. If the criteria is not met (e.g., HIGHER(20) is

specified and the record count is 20 or less), ICETOOL sets RC=0 for the COUNT operator. ICETOOL uses DFSORT's STOPAFT option to process the minimum number of records required to determine whether or not the criteria is met.

RC4 Option Description

RC4 can be used to set RC=4 for this COUNT operator if the record count meets the specified criteria. RC4 can only be specified if EMPTY, NOTEMPTY, HIGHER(x), LOWER(y), EQUAL(v), or NOTEQUAL(w) is specified.

RC4 overrides the default of setting RC=12 for this COUNT operator if the record count meets the specified criteria.

Sample Syntax

```
* Set RC=4 if INPUT1 is empty or
* set RC=0 if INPUT1 is not empty.
 COUNT FROM(INPUT1) EMPTY RC4
```

SSMSG DD

ICETOOL now allows a SSMSG DD statement to be used instead of a DFSMSG DD statement for the DFSORT message data set.

Either a DFSMSG DD statement or an SSMSG DD statement **must** be present. If both are present, ICETOOL uses DFSMSG as the DFSORT message data set. If a DFSMSG DD is not present, but an SSMSG DD is present, ICETOOL uses SSMSG as the DFSORT message data set.

SSMSG should not be used as a ddname in ICETOOL operators.

SFF and UFF formats

The new SFF and UFF formats, described earlier in "Free Form Formats (SFF and UFF)", can be used in the following ICETOOL operators:

- **DISPLAY:** You can use 1-44 bytes for SFF and UFF fields in the ON and BREAK parameters. A maximum of 31 digits is allowed. If a value with more than 31 digits is found, ICETOOL issues an error message and terminates the operation.
- **OCCUR:** You can use 1-44 bytes for SFF and UFF fields in the ON parameter. A maximum of 31 digits is allowed. If a value with more than 31 digits is found, ICETOOL issues an error message and terminates the operation.
- **RANGE:** You can use 1-44 bytes for SFF and UFF fields in the ON parameter. A maximum of 31 digits is allowed. If a value with more than 31 digits is found, ICETOOL issues an error message and terminates the operation.
- **SELECT:** You can use 1-44 bytes for SFF and UFF fields in the ON parameter. A maximum of 44 digits is allowed.
- **SPLICE:** You can use 1-44 bytes for SFF and UFF fields in the ON parameter. A maximum of 44 digits is allowed.
- **STATS:** You can use 1-44 bytes for SFF and UFF fields in the ON parameter. A maximum of 31 digits is allowed. If a value with more than 31 digits is found, ICETOOL issues an error message and terminates the operation.

- **UNIQUE:** You can use 1-44 bytes for SFF and UFF fields in the ON parameter. A maximum of 44 digits is allowed.

Sample Syntax

```
* Select first duplicate records based on
* numeric values with digits and other characters.
  SELECT FROM(IN) TO(OUT) ON(128,40,SFF) FIRSTDUP
```

Larger ZD, PD, FS, BI and FI fields

ICETOOL can now handle larger fields in the ON(p,m,f) and ON(p,m,f,formatting) parameters of the DISPLAY, OCCUR, RANGE, SELECT, SPLICE, STATS, UNIQUE and VERIFY operators, and in the BREAK(p,m,f) and BREAK(p,m,f,formatting) parameters of the DISPLAY operator.

Note: See the discussion of the ON and BREAK parameters in *z/OS DFSORT Application Programming Guide* for the details to which the **changes** discussed in this section apply.

The limits have been changed to allow:

- up to 31 digits for ZD, PD and FS fields (e.g., p,m,ZD)
- up to 8 bytes (20 digits) for BI fields (p,m,BI)
- up to 8 bytes (19 digits) for FI fields (p,m,FI)

The old and new lengths for these fields are as follows:

Item	Old length	New length
ZD field	1-15	1-31
PD field	1-8	1-16
FS field	1-16	1-32
BI field	1-4	1-8
FI field	1-4	1-8

Additionally:

- an edit pattern (E'pattern') can now accomodate up to 44 characters with up to 31 digits.
- dd in Ndd and Udd can be 01 to 31.

For FS, SFF and UFF fields with DISPLAY, OCCUR, RANGE and STATS, a maximum of 31 digits is allowed. If a value with more than 31 digits is found, ICETOOL issues an error message and terminates the operation.

The default number of digits (d) for a TOTAL or BTOTAL with DISPLAY is determined from the format and length of the ON field as follows:

```
Format        Length      needed (d)
--------      ------      ----------
ZD            1-15        15
ZD            16-31       31
PD            1-8         15
PD            9-16        31
FS            1-15        15
FS            16-32       31
SFF, UFF      1-15        15
SFF, UFF      16-44       31
BI, FI        1-4         15
BI, FI        5-8         31
```

For STATS, a new **LMSG** option can be used to specify that the minimum, maximum, average and total for all numeric fields are to be printed using messages that display 31 digits (overriding the default of printing messages that display 15 digits when possible). LMSG ensures that only message ICE648I is used to display the statistics. Without LMSG, a combination of messages ICE608I, ICE609I and ICE648I can be used to display the statistics

Sample Syntax

```
* Display large BREAK and ON fields.
  DISPLAY FROM(IN1) LIST(RPT) ON(28,6,FI) ON(1,23,FS)-
    BTITLE('Break value:') BREAK(39,31,ZD) -
    BTOTAL('Break total:')
* Display minimum, maximum, average and total for
* large and small ON fields as 31 digits.
  STATS FROM(IN2) ON(1,5,PD) ON(21,12,PD) ON(7,8,BI) LMSG
```

Larger constants (RANGE)

DFSORT now allows up to 31 digits (was 15) for decimal constants (n, +n, -n) in the following RANGE operator options:

- HIGHER(x)
- LOWER(x)
- EQUAL(x)
- NOTEQUAL(x)

Sample Syntax

```
  RANGE FROM(INPUT) ON(31,18,ZD) -
   HIGHER(-200000000000000000) LOWER(+200000000000000000)
```

DCn, TCn, DEn and TEn formats (DISPLAY, OCCUR)

The new DC1-3, TC1-4, DE1-3 and TE1-4 formats, described earlier in "TOD and ETOD formats (DCn, TCn, DEn, TEn)", can be used in the ON parameter of ICETOOL's DISPLAY and OCCUR operators, and in the BREAK parameter of ICETOOL's DISPLAY operator, to produce meaningful representations of TOD (STCK) and ETOD (STCKE) date and time values.

You must use 8 bytes for DCn, TCn, DEn and TEn fields. All values are treated as positive.

Sample Syntax

```
* Display a TOD date as C'yyyy/mm/dd' and a
* TOD time as 'hh:mm:ss.xx'.
 DISPLAY FROM(IN) LIST(RPT) -
   HEADER('TOD date') ON(26,8,DC1,E'9999/99/99') -
   HEADER('TOD time') ON(26,8,TC4,E'99:99:99.99')
```

YDDD(abc) and YDDDNS(ab) (DISPLAY, OCCUR)

The new YDDD(abc) and YDDDNS(ab) parameters, described earlier in "Year and day separators (YDDD and YDDDNS)", give you another way to insert the current date in the title line of your DISPLAY and OCCUR reports.

Sample Syntax

```
* Display C'ddd-yyyy' in title.
  DISPLAY FROM(IN) LIST(OUT) TITLE('Division Report') YDDD(D4-) -
    HEADER('Division') HEADER('Revenue') -
    ON(5,8,CH)         ON(16,6,PD,F1)
```

SORT and MERGE Enhancements

SFF and UFF formats

DFSORT's new SFF and UFF formats, described earlier in "Free Form Formats (SFF and UFF)", can be used in SORT and MERGE control fields. You can use 1-44 bytes for SFF and UFF control fields. A maximum of 44 digits is allowed.

Sample Syntax

```
* Sort descending on date values like this:
* yyyy-mm-dd
* yyyy/mm/dd
* and ascending on numeric values like this:
* (d,ddd,ddd.dd)
*  d,ddd,ddd.dd
  SORT FIELDS=(21,10,UFF,D,5,14,SFF,A)
```

Larger FS fields

DFSORT can now handle larger FS control fields for SORT and MERGE. You can use 1-32 bytes (was 1-16) for FS control fields. A maximum of 32 digits is allowed.

```
  SORT FORMAT=FS,FIELDS=(41,32,D,5,18,A)
```

Signs for CSL, CST, ASL and AST fields

DFSORT no longer requires that '+' be used as a positive sign for CSL, CST, ASL and AST control fields for SORT and MERGE. For CSL and ASL values, DFSORT now treats a leading '-' as a negative sign and any other leading character (e.g., blank) as a positive sign. For CST and AST values, DFSORT now treats a trailing '-' as a negative sign and any other trailing character (e.g., blank) as a positive sign.

Sample Syntax

```
* Sort ascending for values like this:
* +ddddd
*  ddddd
* -ddddd
  SORT FIELDS=(45,6,CSL,A)
```

INCLUDE and OMIT Enhancements

SFF and UFF formats

DFSORT's new SFF and UFF formats, described earlier in "Free Form Formats (SFF and UFF)", can be used in INCLUDE, OMIT, OUTFIL INCLUDE, OUTFIL OMIT, and IFTHEN WHEN compare fields. You can use 1-44 bytes for SFF and UFF compare fields. A maximum of 44 digits is allowed.

You can compare SFF fields to decimal constants, and to SFF, UFF, FS, CSL and CST fields.

You can compare UFF fields to decimal constants, and to SFF, UFF, FS, CSL and CST fields.

Sample Syntax

```
* Keep records in which a value in any of these forms:
* (ddd)
* *ddd*
* ddd
*  ddd
*   ddd
* is equal to 408, or a C'dd,ddds'
* value is equal to an sddddd value.
  INCLUDE COND=(51,5,UFF,EQ,+408,OR,
    5,7,SFF,EQ,14,6,FS)
```

Larger FS fields

DFSORT can now handle larger FS compare fields for INCLUDE, OMIT, OUTFIL INCLUDE, OUTFIL OMIT, and IFTHEN WHEN. You can use 1-32 bytes (was 1-16) for FS compare fields. A maximum of 32 digits is allowed.

Sample Syntax

```
  OUTFIL OMIT=(11,20,FS,GT,42,20,FS)
```

Larger constants for FI and BI fields

DFSORT now has larger limits for decimal constants to be compared to FI or BI fields for INCLUDE, OMIT, OUTFIL INCLUDE, OUTFIL OMIT, and IFTHEN WHEN, as follows:

- When an FI field is compared with a decimal constant, n or +n cannot be larger than +9223372036854775807 (was +2147483647) and -n cannot be smaller than -9223372036854775808 (was -2147483648).
- When a BI field is compared with a decimal constant, n or +n cannot be larger than +18446744073709551615 (was +4294967295) nor smaller than +0.

Sample Syntax:

```
  INCLUDE COND=(27,8,FI,GT,-5000000000000000000)
```

SUM Enhancement

Larger ZD fields

DFSORT can now handle larger ZD summary fields for SUM. You can use 1-31 bytes (was 1-18) for ZD summary fields.

Sample Syntax:

```
SUM FIELDS=(5,20,ZD,41,31,ZD)
```

Labels

General Description

DFSORT now allows control statements in SYSIN and SORTCNTL to contain labels up to 70 characters, and allows any character in the label. DFSORT now ignores statements with a label followed only by blanks.

Label Field Description

A label can be specified on any control statement in SYSIN or SORTCNTL. A label is never required. If present, a label must begin in column 1 with any character other than a blank or asterisk (*). A label can be 1 to 70 characters and ends when a blank character is found. Any character can be used in a label. A label followed only by blanks is printed but otherwise not processed.

Labels cannot be specified in the parameter list, in DFSPARM or in continuation lines.

To skip the label, specify one or more blanks starting in column 1.

Sample Syntax

```
* No label for the OPTION statement.
  OPTION EQUALS
* MY*SORT*CARD is the label for the SORT statement.
MY*SORT*CARD SORT FIELDS=(5,4,CH,A)
* No label for the OUTREC statement.
   OUTREC FIELDS=(1,20,51,30)
* OUT_1 is the label for the first OUTFIL statement.
OUT_1  OUTFIL FNAMES=OUT1,INCLUDE=(5,1,CH,EQ,C'A')
* OUT_2 is the label for the second OUTFIL statement.
OUT_2  OUTFIL FNAMES=OUT2,INCLUDE=(5,1,CH,EQ,C'B')
```

LRECL in concatenation

For sort and copy applications, with concatenated variable-length input data sets for SORTIN, DFSORT now uses the largest LRECL it finds in the concatenation.

The following rules apply to concatenated data sets for SORTIN:

- With fixed-length records, LRECL must be the same for all data sets.

With variable-length records, LRECL can vary. However:

- If Blockset is selected: If a tape data set has the largest LRECL and is not first in the concatenation, you must specify LRECL explicitly on its DD statement if the LRECL is not available from DFSMSrmm(tm) or ICETPEX.
- If Blockset is not selected, the first data set in the concatenation must have the largest LRECL (LRECL can be specified explicitly on its DD statement).

Sample SORTIN

```
//* DFSORT uses the maximum LRECL of 150 from
//* the third data set in the concatenation.
//SORTIN DD DSN=XYZ.VB100,DISP=SHR
//       DD DSN=XYZ.VB90,DISP=SHR
//       DD DSN=XYZ.VB150,DISP=SHR
//       DD DSN=XYZ.VB120,DISP=SHR
```

Symbols Enhancements

SFF and UFF formats

DFSORT's new SFF and UFF formats, described earlier in "Free Form Formats (SFF and UFF)", can be used as the format (f) in symbols for fields in the same way that the other formats can be used. You can specify f using uppercase letters (e.g., SFF), lowercase letters (e.g., uff) or mixed case letters (e.g., Sff). f specified in any case will be treated like uppercase.

Note: SFF and UFF are allowed as formats, but not allowed as symbols (e.g., Sym1,5,12,SFF is allowed, but SFF,5,12,SFF is not). However, lower case and mixed case forms of these words, such as Sff or uff, can be used as symbols (e.g., sff,5,12,SFF is allowed).

Sample Syntax

```
//SYMNAMES DD *
Sym1,11,12,SFF
Unsigned-Free-Form,55,20,uff
/*
//SYSIN DD *
  SORT FIELDS=(Sym1,A,Unsigned-Free-Form,D)
/*
```

Larger decimal constants

DFSORT now allows symbols for decimal constants (n, +n or -n) to contain from 1-31 significant digits (was 1-15).

Sample Syntax

```
//SYMNAMES DD *
Con1,+12345678901234567890
Large_decimal_number,-50000000000000000000000000000000
Num1,21,20,ZD
Num2,51,44,SFF
/*
//TOOLIN DD *
  RANGE FROM(IN1) ON(Num1) LOWER(Con1)
  RANGE FROM(IN2) ON(Num2) HIGHER(Large_decimal_number)
/*
```

DCn, TCn, DEn and TEn formats

DFSORT's new DC1-3, TC1-4, DE1-3 and TE1-4 formats, described earlier in "TOD and ETOD formats (DCn, TCn, DEn, TEn)", can be used as the format (f) in symbols for fields in the same way that the other formats can be used. You can specify f using uppercase letters (e.g., DC2), lowercase letters (e.g., te1) or mixed case letters (e.g., De3). f specified in any case will be treated like uppercase.

Note: DC1, DC2, DC3, TC1, TC2, TC3, TC4, DE1, DE2, DE3, TE1, TE2, TE3 and TE4 are allowed as formats, but not allowed as symbols (e.g., Sym1,5,8,DC2 is allowed, but DC2,5,8,DC2 is not). However, lower case and mixed case forms of these words, such as dc2 or Te1, can be used as symbols (e.g., dc2,5,8,DC2 is allowed).

Sample Syntax

```
//SYMNAMES DD *
Sym1,11,8,DE3
TOD-time,55,8,TC4
/*
//SYSIN DD *
  OPTION COPY
  OUTREC BUILD=(C'Date is ',Sym1,EDIT=(TTTT/TTT),
    C' , time is ',TOD-time,M11)
/*
```

New and Changed Messages

This section shows messages that have been added, or changed significantly, for PTFs UQ95214 and UQ95213. Refer to *z/OS DFSORT Messages, Codes and Diagnosis Guide* for general information on DFSORT messages.

ICE012A

ICE012A MISSING FIELDS OPERAND DEFINER

Explanation: Critical. A SORT, MERGE, or SUM control statement did not contain the FIELDS operand.

System Action: The program terminates.

Programmer Response: Check for a SORT, MERGE, or SUM control statement that does not have the FIELDS operand.

ICE018A

ICE018A INVALID OR MISSING FORMAT

Explanation: Critical. This message was issued for one of the following reasons:

1. More than 112 control fields were specified, and Blockset was not selected.
2. A SORT, MERGE, OUTFIL, SUM, INCLUDE, or OMIT statement contained an invalid format type. For example:

```
   SORT FIELDS=(5,4,NG,A)
*  NG IS AN INVALID FORMAT

   SUM FIELDS=(12,2),FORMAT=CH
*  CH IS AN INVALID FORMAT FOR SUM
```

3. For a SORT or MERGE control statement, the format was invalid for the length specified.
4. CSF, FS, UFF, SFF, Y2x or PD0 format was specified for SORT or MERGE and Blockset was not selected.
5. D2 format was specified in the INCLUDE or OMIT operand of an OUTFIL statement, or in the IFTHEN WHEN operand of an INREC, OUTREC or OUTFIL statement.
6. The INCLUDE or OMIT operand of an OUTFIL statement, or an IFTHEN WHEN operand of an INREC, OUTREC or OUTFIL statement, contained an invalid format type or a field without a format. For example:

```
  OUTFIL FNAMES=OUT1,
    INCLUDE=(5,2,NG,EQ,C'AB')
*     NG IS AN INVALID FORMAT

  OUTFIL FNAMES=OUT2,

    IFTHEN=(WHEN=(21,2,EQ,35,2),OVERLAY=(21:C'OK'))
*     FORMATS ARE MISSING IN WHEN CONDITION
```

7. A SORT, MERGE, SUM, INCLUDE, or OMIT statement without a FORMAT=f operand contained a field without a format (that is, p,m instead of p,m,f). For example:

```
  SORT FIELDS=(5,4,BI,A,21,2,D)
*  FORMAT IS MISSING FOR SECOND FIELD
```

System Action: The program terminates.

Programmer Response:

For cases 1 through 3, check that each format type is valid for the control statement specified.

For case 4, rerun the job with a SORTDIAG DD statement to get message ICE800I, which indicates the reason Blockset could not be used. If possible, remove the condition preventing the use of Blockset.

For case 5, use an INCLUDE or OMIT statement, or change the D2 format to a format that is valid for the INCLUDE or OMIT operand of an OUTFIL statement, or for the IFTHEN WHEN operand of an INREC, OUTREC or OUTFIL statement, as appropriate.

For case 6, check that each field is specified as p,m,f with a valid format for f. For example:

```
OUTFIL FNAMES=OUT1,
  INCLUDE=(5,2,CH,EQ,C'AB')

OUTFIL FNAMES=OUT2,

  IFTHEN=(WHEN=(21,2,BI,EQ,35,2,BI),OVERLAY=(21:C'OK'))
```

For case 7, check that each field is specified as p,m,f or that a FORMAT=f operand is specified. For example:

```
SORT FIELDS=(5,4,BI,A,21,2,D),FORMAT=FI
```

ICE107A

ICE107A DUPLICATE, CONFLICTING, OR MISSING INREC OR OUTREC STATEMENT OPERAND

Explanation: Critical. One of the following errors was found in an INREC or OUTREC statement:

- FIELDS, BUILD or OVERLAY operand was specified twice.
  Example:

  ```
  INREC FIELDS=(5,4,C'***',40:X),FIELDS=(1,60)
  ```

- FIELDS and BUILD, FIELDS and OVERLAY, FIELDS and IFTHEN, BUILD and OVERLAY, BUILD and IFTHEN, or OVERLAY and IFTHEN were specified.
  Example:

  ```
  OUTREC BUILD=(1,20),OVERLAY=(10:C'A')
  ```

- IFOUTLEN and BUILD, IFOUTLEN and FIELDS or IFOUTLEN and OVERLAY were specified.
  Example:

  ```
  INREC OVERLAY=(21:C'A'),IFOUTLEN=50
  ```

- For an IFTHEN clause, WHEN was not specified.
  Example:

  ```
  OUTREC IFTHEN=(OVERLAY=(10:C'A'))
  ```

- For an IFTHEN clause, WHEN=INIT, WHEN=(logexp) or WHEN=NONE was specified without BUILD or OVERLAY.
  Example:

  ```
      INREC IFTHEN=(WHEN=(5,1,CH,EQ,C'1'))
  ```

- An IFTHEN clause with WHEN=INIT was preceded by an IFTHEN clause with WHEN=(logexp), WHEN=ANY or WHEN=NONE.
  Example:

  ```
  OUTREC IFTHEN=(WHEN=(5,2,CH,EQ,C'AA'),OVERLAY=(10:C'A')),
         IFTHEN=(WHEN=INIT,BUILD=(1,80))
  ```

- An IFTHEN clause with WHEN=NONE was followed by an IFTHEN clause with WHEN=INIT, WHEN=(logexp) or WHEN=ANY.
  Example:

  ```
  INREC IFTHEN=(WHEN=NONE,OVERLAY=(10:C'A')),
        IFTHEN=(WHEN=ANY,BUILD=(1,80))
  ```

- The first IFTHEN clause with WHEN=ANY was not preceded by an IFTHEN clause with WHEN=(logexp).
  Example:

  ```
  OUTREC IFTHEN=(WHEN=INIT,OVERLAY=(10:C'A')),
         IFTHEN=(WHEN=ANY,BUILD=(1,80))
  ```

- An IFTHEN clause with WHEN=ANY and without HIT=NEXT was followed by an IFTHEN clause with WHEN=ANY. Example:
  Example:

```
OUTREC IFTHEN=(WHEN=(5,1,CH,EQ,C'1'),OVERLAY=(10:C'A'),HIT=NEXT),
       IFTHEN=(WHEN=(5,1,CH,EQ,C'2'),OVERLAY=(10:C'B'),HIT=NEXT),
       IFTHEN=(WHEN=ANY,OVERLAY=(28:C'ABC')),
       IFTHEN=(WHEN=ANY,BUILD=(1,80))
```

System Action: The program terminates.

Programmer Response: Check the INREC or OUTREC control statement for the errors indicated in the explanation and correct the errors.

ICE109A

ICE109A SUM FIELD DISPLACEMENT OR LENGTH VALUE ERROR

Explanation: Critical. A sum field definition on a SUM control statement contained an invalid length or displacement (position) value.

System Action: The program terminates.

Programmer Response: Make sure that the length and position values in the FIELDS operand of the SUM control statement were specified correctly. For BI and FI, length must be 2, 4, or 8 bytes; for PD, length must be 1 through 16 bytes; for ZD, length must be 1 through 31 bytes; for FL, length must be 4, 8 or 16 bytes. Make sure that the position plus length of each field does not exceed 4093.

ICE111A

ICE111A REFORMATTING FIELD ERROR

Explanation: Critical. The FIELDS, BUILD, OVERLAY, IFTHEN BUILD or IFTHEN OVERLAY operand of an INREC or OUTREC statement, or the OUTREC, BUILD, OVERLAY, IFTHEN BUILD or IFTHEN OVERLAY operand of an OUTFIL statement, contained an invalid column, separator, position, length, keyword, pattern, sign, constant or value. Some common errors are:

- A 0 value was used.
- A null value was used where it was not permitted.
- A null string, pattern, or sign was used.
- A column was greater than 32752, or was followed by another column.
- A position plus length was greater than 32753.
- DATE=(abcd) or DATENS(abc) was specified with a, b or c not M, D, Y or 4, with M, D, Y or 4 specified more than once, or with Y and 4 both specified.
- YDDD=(abc) or YDDDNS=(ab) was specified with a or b not D, Y or 4, with D, Y or 4 specified more than once, or with Y and 4 both specified.
- TIME=(abc) or TIMENS=(ab) was specified with ab not 12 or 24.
- The length for a hexadecimal field was greater than 16376.
- A repetition factor was greater than 4095 for a separator, or a character or hex constant was longer than 256 bytes.

- An invalid digit or an odd number of digits was specified for a hexadecimal constant.
- The length for a Y2 format field was not 2 for Y2C, Y2Z, Y2P, Y2S or Y2PP, or 1 for Y2D, Y2B or Y2DP, or 3-6 for Y2T, Y2W, Y2TP or Y2WP, or 2-3 for Y2U, Y2X, Y2UP or Y2XP, or 3-4 for Y2V, Y2Y, Y2VP or Y2YP.
- The length for an edit field was less than 2 bytes or greater than 8 bytes for PD0, or was greater than 8 bytes for BI or FI, 16 bytes for PD, 31 bytes for ZD, 32 bytes for CSF/FS, 44 bytes for UFF or SFF, or was not 4 bytes for DT1, DT2, DT3, TM1, TM2, TM3, or TM4, or 8 bytes for DC1, DC2, DC3, TC1, TC2, TC3, TC4, DE1, DE2, DE3, TE1, TE2, TE3, or TE4.
- More than 31 digits or 44 characters were specified in an edit pattern.
- SIGNz (where z is not S) was specified with Mn or without EDIT or EDxy.
- x, y, or z in EDxy or SIGNz were the same character.
- The value for LENGTH was greater than 44.
- NOMATCH was specified after position and length rather than after position, length, and CHANGE.
- The length for a lookup input field was greater than 64 with character or hexadecimal find constants, or greater than 1 with find bit constants.
- The length for a lookup output field was greater than 64.
- The length for a find constant was greater than the lookup input field length.
- A find constant was not a character, hexadecimal, or bit constant.
- The length for a set constant or set field was greater than the lookup output field length
- An invalid character was specified in a find bit constant, or the number of bits for a find bit constant was not 8.
- A set constant was not a character or hexadecimal constant.
- The length for a sequence number was greater than 16.
- The value for START was greater than 100000000000.
- The value for INCR was greater than 10000000.
- The length for a RESTART field was greater than 256 bytes.
- A position without length (p without m) was specified for OVERLAY or IFTHEN OVERLAY.
- A / was specified for OVERLAY or IFTHEN OVERLAY.

System Action: The program terminates.

Programmer Response: Correct the invalid value.

ICE113A

ICE113A COMPARISON FIELD ERROR

Explanation: Critical. The COND operand of an INCLUDE or OMIT statement, or the INCLUDE or OMIT operand of an OUTFIL statement, or the IFTHEN WHEN operand of an INREC, OUTREC or OUTFIL statement, contained an invalid position, length, format, constant or mask.

System Action: The program terminates.

Programmer Response: Make sure that all fields and constants are specified correctly. Make sure that the position plus the length of each field does not exceed 32753. Make sure that a shift-out (X'0E') is not embedded within double-byte data in a character constant .

ICE114A

ICE114A INVALID COMPARISON

Explanation: Critical. One of the following situations exists:

- The COND operand of an INCLUDE or OMIT statement, or the INCLUDE or OMIT operand of an OUTFIL statement, or the IFTHEN WHEN operand of an INREC, OUTREC or OUTFIL statement, contained an invalid field-to-field, field-to-mask, or field-to-constant comparison.
- FORMAT=SS was specified after COND in an INCLUDE or OMIT statement.
- Locale processing was required, but a character (CH) field to binary (BI) field comparison was specified. Locale processing does not support the comparison of a CH to BI field.

System Action: The program terminates.

Programmer Response: Make sure that all of the comparisons are valid. If you specified FORMAT=SS after COND in an INCLUDE or OMIT statement, respecify it before COND. DFSORT's locale processing might eliminate the need for CH to BI comparisons. See *z/OS DFSORT Application Programming Guide* for information relating to locale processing. If a CH to BI comparison is needed, specify run time option LOCALE=NONE.

ICE126A

ICE126A INCONSISTENT {*INREC|*OUTREC|ddname} IFTHEN n REFORMATTING FIELD FOUND

Explanation: Critical. The FIELDS, BUILD or IFTHEN BUILD operand of an INREC or OUTREC statement, or the OUTREC, BUILD or IFTHEN BUILD operand of an OUTFIL statement, contained a field that was inconsistent with the format (fixed or variable) of the input or output records, or with other fields.

OUTFIL data sets for which VTOF or CONVERT is used have variable-length input records and fixed length output records. OUTFIL data sets for which FTOV is used have fixed-length input records variable-length output records.

The specific cause of the error is indentified as follows:

- *INREC and n=0 indicates that the FIELDS or BUILD operand of the INREC statement caused the error.
- *OUTREC and n=0 indicates that the FIELDS or BUILD operand of the OUTREC statement caused the error.
- ddname and n=0 indicates that the OUTREC or BUILD operand of an OUTFIL statement caused the error. ddname identifies the first data set in the associated OUTFIL group.
- *INREC and n>0 indicates that an IFTHEN BUILD operand of the INREC statement caused the error. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the INREC statement).
- *OUTREC and n>0 indicates that an IFTHEN BUILD operand of the OUTREC statement caused the error. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the OUTREC statement).

- ddname and n>0 indicates that an IFTHEN BUILD operand of an OUTFIL statement caused the error. ddname identifies the first data set in the associated OUTFIL group. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the OUTFIL statement).

The inconsistency is one of the following:

1. The first field was not 1,n where n is equal to or greater than 4, for variable-length input records, that is, the first field did not contain the record descriptor word (RDW).
   Example (variable-length input:):

   ```
   INREC FIELDS=(6,8)
   ```

2. The first field was 1,n, but was modified in some way (for example, 1,n,f or 1,n,TRAN=UTOL), for variable-length input records.
   Example (variable-length input):

   ```
   INREC BUILD=(1,6,HEX)
   ```

3. A single field containing only bytes from the RDW was specified for variable-length records.
   Example (variable-length input):

   ```
   OUTREC BUILD=(1,4)
   ```

4. The last position was specified without a corresponding length for fixed-length input records.
   Example (fixed-length input):

   ```
   OUTFIL IFTHEN=(WHEN=(5,1,CH,EQ,C'A'),BUILD=(1,5,8)),
          IFTHEN=(WHEN=NONE,OVERLAY=(21:C'OK'))
   ```

5. For variable-length input records, the last position was specified without a corresponding length for one, but not all, of the FIELDS or BUILD operand of the INREC statement, the FIELDS or BUILD operand of the OUTREC statement, or the OUTREC or BUILD operand of an OUTFIL statement (without a VTOF or CONVERT operand):
   Example (variable-length input):

   ```
   INREC FIELDS=(1,20,31)
   OUTREC FIELDS=(1,25)
   ```

6. For variable-length input records, the last position was specified with a corresponding length for one, but not all, of the FIELDS or BUILD operand of the INREC statement, the FIELDS or BUILD operand of the OUTREC statement, or the OUTREC or BUILD operand of an OUTFIL statement (without a VTOF or CONVERT operand):
   Example (variable-length input):

   ```
   OUTREC BUILD=(1,20,31,10)
   OUTFIL BUILD=(1,25,27)
   ```

7. A column overlapped the previous output field in the reformatted record.
   Example (fixed-length input):

   ```
   INREC FIELDS=(3,20,20:C'ABC')
   ```

8. A VTOF or CONVERT operand was specified without an OUTREC or BUILD operand for variable-length input record to fixed-length output record conversion in an OUTFIL statement.
   Example (variable-length input):

   ```
   OUTFIL VTOF or
   OUTFIL VTOF,OVERLAY=(21:C'A')
   ```

9. A VLFILL = C'x' or VLFILL = X'yy' operand was specified without an OUTREC or BUILD operand for variable-length input records in an OUTFIL statement.
   Example (variable-length input):

   ```
   OUTFIL VLFILL=C'*'
   ```

10. An OUTREC or BUILD operand did not specify any input or separation fields.
    Example (fixed-length input):

    ```
    OUTFIL OUTREC=(//)
    ```

11. An IFOUTLEN=n operand was specified without an IFTHEN operand.
    Example (fixed-length input):

    ```
    OUTREC IFOUTLEN=50
    ```

System Action: The program terminates.

Programmer Response:

1. For case 1, specify 1,n for the first field, where n is equal to or greater than 4.
   Example (variable-length input):

   ```
   INREC FIELDS=(1,4,6,8)
   ```

2. For case 2, specify 1,4 before the modified field
   Example (variable-length input):

   ```
   INREC BUILD=(1,4,1,6,HEX)
   ```

3. For case 3, specify at least one data byte.
   Example (variable-length input):

   ```
   OUTREC BUILD=(1,5)
   ```

4. For case 4, specify the last position with a corresponding length.
   Example (fixed-length input):

   ```
   OUTFIL IFTHEN=(WHEN=(5,1,CH,EQ,C'A'),BUILD=(1,5,8,25)),
     IFTHEN=(WHEN=NONE,OVERLAY=(21:C'OK'))
   ```

5. For case 5, specify the last position without a corresponding length for the INREC, OUTREC, and OUTFIL statements, as appropriate.
   Example (variable-length input):

   ```
   INREC FIELDS=(1,20,31)
   OUTREC FIELDS=(1,20,5X,21)
   ```

6. For case 6, specify the last position with a corresponding length for the INREC, OUTREC, and OUTFIL statements, as appropriate.
   Example (variable-length input):

   ```
   OUTREC BUILD=(1,20,31,10)
   OUTFIL BUILD=(1,20,5X,21,10)
   ```

7. For case 7, ensure that columns do not overlap previous fields in the reformatted record.
   Example (fixed-length input):

   ```
   INREC FIELDS=(3,20,25:C'ABC')
   ```

8. For case 8, specify an OUTREC or BUILD operand with the VTOF or CONVERT operand.
   Example (variable-length input):

   ```
   OUTFIL VTOF,BUILD=(5,60)
   ```

9. For case 9, specify an OUTREC or BUILD operand with the VLFILL=C'x' or VLFILL=X'yy' operand.
   Example (variable-length input):

   ```
   OUTFIL VLFILL=C'*',BUILD=(1,4,8,45)
   ```

10. For case 10, specify an input or separation field.
    Example (fixed-length input):

```
OUTFIL OUTREC=(//,X)
```

11. For case 11, specify an IFTHEN operand with an IFOUTLEN=n operand.
    Example:

```
OUTREC IFOUTLEN=50,
  IFTHEN=(WHEN=(5,1,CH,EQ,C'A'),OVERLAY=(21:C'N')),
  IFTHEN=(WHEN=NONE,OVERLAY=(21:C'Y'))
```

ICE151A

ICE151A TOO MANY {*INCLUDE|*OMIT|*INREC|*OUTREC|ddname} IFTHEN n CONDITIONS

Explanation: Critical. The complexity of the conditions in a COND, INCLUDE, OMIT or IFTHEN WHEN operand caused dynamic areas to exceed the storage allowed for them. The specific cause of the error is identified as follows:

- *INCLUDE indicates that the COND operand of the INCLUDE statement caused the error. n is 0.
- *OMIT indicates that the COND operand of the OMIT statement caused the error. n is 0.
- ddname and n=0 indicates that the INCLUDE or OMIT operand of an OUTFIL statement caused the error. ddname identifies the first data set in the associated OUTFIL group.
- *INREC indicates that an IFTHEN WHEN operand of the INREC statement caused the error. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the INREC statement). - *OUTREC indicates that an IFTHEN WHEN operand of the OUTREC statement caused the error. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the OUTREC statement).
- ddname and n>0 indicates that an IFTHEN WHEN operand of an OUTFIL statement caused the error. ddname identifies the first data set in the associated OUTFIL group. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the OUTFIL statement).

System Action: The program terminates.

Programmer Response: Reduce the number of conditions, or the size of the constants, in the COND, INCLUDE, OMIT or IFTHEN WHEN operand that caused the problem. Alternatively, you may be able to avoid reducing the number of conditions, or the size of the constants, by using one of the following techniques:

- If NOSZERO is in effect and you can treat numeric -0 and +0 values as signed (that is, different) for this application, use the SZERO option.
- Rewrite conditions to use substring comparison tests (see *z/OS DFSORT Application Programming Guide* for details), if possible.
- Use an INREC or OUTREC statement with multiple IFTHEN clauses to "OR" sets of conditions together and set a "flag" that an OUTFIL statement can use to delete the appropriate records. In the following example, the input data set has RECFM=FB and LRECL=80, and a1 to an and b1 to bn represent relational conditions:

```
//MULT  EXEC  PGM=ICEMAN
//SYSOUT    DD  SYSOUT=*
//SORTIN DD DSN=...
//SORTOUT DD DSN=...
//SYSIN    DD    *
  OPTION COPY
* If a1,OR,a2,OR,...,an is true,
* set flag in 81 to 'Y'.
  INREC IFTHEN=(WHEN=(a1,OR,a2,OR,...,an),
          OVERLAY=(81:C'Y')),
* If b1,OR,b2,OR,...,bn is true,
* set flag in 81 to 'Y'.
     IFTHEN=(WHEN=(b1,OR,b2,OR,...,bn),
       OVERLAY=(81:C'Y')),
* If none of the conditions is true,
* set flag in 81 to 'N'.
     IFTHEN=(WHEN=NONE,OVERLAY=(81:C'N'))
* If the flag in 81 is 'Y', include the
* records.
  OUTFIL INCLUDE=(81,1,CH,EQ,C'Y'),
* Remove the flag byte.
    OUTREC=(1,80)
/*
```

ICE189A

ICE189A BLOCKSET REQUIRED BUT COULD NOT BE USED - REASON CODE IS nn

Explanation: Critical. Blockset was required for one of the following:

- LOCALE processing
- OUTFIL processing
- Y2x, Y2xx, PD0, FS, CSF, UFF or SFF format
- OVERLAY, IFTHEN or IFOUTLEN processing
- INREC or OUTREC processing with one of the following:
  - p,m,HEX
  - p,HEX
  - p,m,TRAN=LTOU
  - p,TRAN=LTOU
  - p,m,TRAN =UTOL
  - p,TRAN=UTOL
  - p,m,TRAN=ALTSEQ
  - p,TRAN=ALTSEQ
  - p,m,f
  - p,m,lookup
  - SEQNUM

  - DATE1, DATE1(c), DATE1P, DATE2, DATE2(c), DATE2P, DATE3, DATE3(c), DATE3P, DATE4, DATE, DATE=(abcd), DATENS=(abc), YDDD=(abc) or YDDDNS=(ab)
  - TIME1, TIME1(c), TIME1P, TIME2, TIME2(c), TIME2P, TIME3, TIME3P, TIME, TIME=(abc) or TIMENS=(ab)
  - +n
  - -n
  - (...)
- A VSAM extended addressability data set
- To set the SORTOUT LRECL from the L3 length (without E35, INREC or OUTREC), the OUTREC length or the INREC length, with SOLRF in effect
- an FL format sort field with NOSZERO in effect
- VLLONG in effect and SORTOUT present
- VSAMEMT in effect for a sort or merge with VSAM input
- The same VSAM data set was specified for both input and output
- An HFS file was specified for input or output
- A tape data set with a block size greater than 32760 bytes was specified for input or output
- SDB=LARGE or SDB=INPUT was in effect and DFSORT selected a block size greater than 32760 bytes for a tape output data set
- VLSHRT in effect with a SUM statement
- Position plus length for a control field exceeded 4093
- ICETOOL called DFSORT for an operation involving SORTOUT, and NULLOUT=RC16 is in effect.

However, Blockset could not be used due to the reason indicated by reason code nn. See message ICE800I for the meaning of nn.

System Action: The program terminates.

Programmer Response: Correct the situation indicated by the reason code so Blockset can be used. Alternatively, you can remove the source of the requirement to use Blockset. However, this will result in the use of a less efficient technique.

ICE212A

ICE212A MATCH NOT FOUND FOR {*INREC|*OUTREC|ddname} IFTHEN n CHANGE FIELD AT POSITION p

Explanation: Critical. In the FIELDS, BUILD, OVERLAY, IFTHEN BUILD or IFTHEN OVERLAY operand of an INREC or OUTREC statement, or the OUTREC, BUILD, OVERLAY, IFTHEN BUILD or IFTHEN OVERLAY operand of an OUTFIL statement, a CHANGE parameter was specified without a NOMATCH parameter and an input field value did not match any of the find constants.

The specific cause of the error is indentified as follows:

- *INREC and n=0 indicates that a change field at position p in the FIELDS, BUILD or OVERLAY operand of the INREC statement caused the error.

- *OUTREC and n=0 indicates that a change field at position p in the FIELDS, BUILD or OVERLAY operand of the OUTREC statement caused the error.
- ddname and n=0 indicates that a change field at position p in the OUTREC, BUILD or OVERLAY operand of an OUTFIL statement caused the error. ddname identifies the first data set in the associated OUTFIL group.
- *INREC and n>0 indicates that a change field at position p in an IFTHEN BUILD or IFTHEN OVERLAY operand of the INREC statement caused the error. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the INREC statement).
- *OUTREC and n>0 indicates that a change field at position p in an IFTHEN BUILD or IFTHEN OVERLAY operand of the OUTREC statement caused the error. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the OUTREC statement).
- ddname and n>0 indicates that a change field at position p in an IFTHEN BUILD or IFTHEN OVERLAY operand of an OUTFIL statement caused the error. ddname identifies the first data set in the associated OUTFIL group. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the OUTFIL statement).

System Action: The program terminates when the first input field is encountered for which a match is not found.

Programmer Response: Correct the lookup table specified with the CHANGE parameter, or use the NOMATCH parameter to specify a constant or input field to be used as the output field if a match is not found. The use of a constant such as NOMATCH=(C'**') can be helpful in identifying all input field values for which a match is not found.

ICE214A

ICE214A DUPLICATE, CONFLICTING, OR MISSING OUTFIL STATEMENT OPERANDS

Explanation: Critical. One of the following errors was found in an OUTFIL statement:

- An operand, other than IFTHEN, was specified twice.
  Example:

  ```
  OUTFIL STARTREC=5,STARTREC=10
  ```

- INCLUDE and OMIT, INCLUDE and SAVE, or OMIT and SAVE were specified.
  Example:

  ```
  OUTFIL INCLUDE=ALL,SAVE
  ```

- VTOF and CONVERT were specified.
  Example:

  ```
  OUTFIL VTOF,CONVERT
  ```

- FTOV and VTOF, FTOV and CONVERT, or FTOV and VLFILL were specified.
  Example:

  ```
  OUTFIL FTOV,VLFILL=C'*'
  ```

- OUTREC and BUILD, OUTREC and OVERLAY, OUTREC and IFTHEN, BUILD and OVERLAY, BUILD and IFTHEN, or OVERLAY and IFTHEN were specified.
  Example:

  ```
  OUTFIL BUILD=(1,20),OVERLAY=(10:C'A')
  ```

- IFOUTLEN and BUILD, IFOUTLEN and OUTREC or IFOUTLEN and OVERLAY were specified.
  Example:

  ```
  OUTFIL OVERLAY=(21:C'A'),IFOUTLEN=50
  ```

- For an IFTHEN clause, WHEN was not specified.
  Example:

```
OUTFIL IFTHEN=(OVERLAY=(10:C'A'))
```

- For an IFTHEN clause, WHEN=INIT, WHEN=(logexp) or WHEN=NONE was specified without BUILD or OVERLAY.
  Example:

```
OUTFIL IFTHEN=(WHEN=(5,1,CH,EQ,C'1'))
```

- For an IFTHEN clause, WHEN=INIT and BUILD with / were specified.
  Example:

```
OUTFIL IFTHEN=(WHEN=INIT,BUILD=(1,25,/,26,25))
```

- For an IFTHEN clause, BUILD with / and HIT=NEXT were specified.
  Example:

```
OUTFIL IFTHEN=(WHEN=(21,1,CH,EQ,C'A'),
         BUILD=(1,25,/,26,25),HIT=NEXT)
```

- An IFTHEN clause with WHEN=INIT was preceded by an IFTHEN clause with WHEN=(logexp), WHEN=ANY or WHEN=NONE.
  Example:

```
OUTFIL IFTHEN=(WHEN=(5,2,CH,EQ,C'AA'),OVERLAY=(10:C'A')),
       IFTHEN=(WHEN=INIT,BUILD=(1,80))
```

- An IFTHEN clause with WHEN=NONE was followed by an IFTHEN clause with WHEN=INIT, WHEN=(logexp) or WHEN=ANY.
  Example:

```
OUTFIL IFTHEN=(WHEN=NONE,OVERLAY=(10:C'A')),
       IFTHEN=(WHEN=ANY,BUILD=(1,80))
```

- The first IFTHEN clause with WHEN=ANY was not preceded by an IFTHEN clause with WHEN=(logexp).
  Example:

```
OUTFIL IFTHEN=(WHEN=INIT,OVERLAY=(10:C'A')),
       IFTHEN=(WHEN=ANY,BUILD=(1,80))
```

- An IFTHEN clause with WHEN=ANY and without HIT=NEXT was followed by an IFTHEN clause with WHEN=ANY:
  Example:

```
OUTFIL IFTHEN=(WHEN=(5,1,CH,EQ,C'1'),OVERLAY=(10:C'A'),HIT=NEXT),
       IFTHEN=(WHEN=(5,1,CH,EQ,C'2'),OVERLAY=(10:C'B'),HIT=NEXT),
       IFTHEN=(WHEN=ANY,OVERLAY=(28:C'ABC')),
       IFTHEN=(WHEN=ANY,BUILD=(1,80))
```

System Action: The program terminates.

Programmer Response: Check the OUTFIL control statement for the errors indicated in the explanation and correct the errors.

ICE215A

ICE215A SPECIFIED FEATURE IS ONLY SUPPORTED BY OUTFIL STATEMENT

Explanation: Critical. One of the following, which is only supported by the OUTFIL control statement, was specified on an INREC or OUTREC control statement:

- A / in a FIELDS, BUILD or IFTHEN BUILD operand
- A VTOF or CONVERT operand
- An FTOV operand
- A VLFILL=byte operand
- A VLTRIM=byte operand

System Action: The program terminates.

Programmer Response: If a / was specified, use a / in a BUILD or IFTHEN BUILD operand in an OUTFIL statement instead. If a VTOF, CONVERT, or VLFILL=byte operand was specified, use the same operand and BUILD in an OUTFIL statement instead. For example:

```
OUTFIL BUILD=(1,60),FTOV
```

If an FTOV or VLTRIM=byte operand was specified, use the same operand and BUILD or IFTHEN BUILD in an OUTFIL statement instead. For example:

```
OUTFIL VLTRIM=X'40',
   IFTHEN=(WHEN=(5,1,CH,EQ,C'A'),BUILD=(1,40,45))
```

ICE218A

ICE218A n BYTE VARIABLE RECORD IS SHORTER THAN m BYTE MINIMUM FOR <ddname> FIELDS

Explanation: Critical. A variable-length record was shorter than 4 bytes, or was too short to contain all specified fields. The values shown in the message are as follows:

- n is the length of the variable-length record
- m is 4 or the minimum length needed for all fields specified
- if ddname is blank, the short record was detected during non-OUTFIL processing. Otherwise, the first data set in the OUTFIL group for which the short record was detected is indicated by ddname.

System Action: The program terminates.

Programmer Response: If a variable-length record was too short to contain all INREC, OUTREC, or OUTFIL fields, use an INREC or OUTREC statement with operands OVERLAY or IFTHEN as appropriate, or an OUTFIL statement with operands OVERLAY, IFTHEN, or BUILD and VLFILL=C 'x' or VLFILL=X 'yy' as appropriate, to prevent DFSORT from terminating.

If the variable-length record was too short to contain all SORT, MERGE, or SUM fields, use the VLSHRT option to prevent DFSORT from terminating.

If a variable-length record was too short to contain all INCLUDE or OMIT fields, use the VLSCMP or VLSHRT option to prevent DFSORT from terminating.

See the **Programmer Response** for message ICE015A for additional information.

ICE221A

ICE221A INVALID FIELD OR CONSTANT IN {*INCLUDE|*OMIT|*INREC|*OUTREC|ddname} IFTHEN n CONDITION m

Explanation: An error was detected in a COND, INCLUDE, OMIT or IFTHEN WHEN operand. The specific cause of the error is identified as follows:

- *INCLUDE indicates that the COND operand of the INCLUDE statement caused the error. n is 0.
- *OMIT indicates that the COND operand of the OMIT statement caused the error. n is 0.
- ddname and n=0 indicates that the INCLUDE or OMIT operand of an OUTFIL statement caused the error. ddname identifies the first data set in the associated OUTFIL group.
- *INREC indicates that an IFTHEN WHEN operand of the INREC statement caused the error. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the INREC statement).
- *OUTREC indicates that an IFTHEN WHEN operand of the OUTREC statement caused the error. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the OUTREC statement).
- ddname and n>0 indicates that an IFTHEN WHEN operand of an OUTFIL statement caused the error. ddname identifies the first data set in the associated OUTFIL group. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the OUTFIL statement).

One of the following errors was detected:

- the length for a field with a format other than SS was greater than 256
- the length for a PD field was 256
- the length for a PD0 field was less than 2 or greater than 8
- the length for a CSF or FS field was greater than 32
- the length for a UFF or SFF field was greater than 44
- the length for a CSL, CST, ASL, or AST field was 1
- the decimal constant for an FI field was greater than +9223372036854775807 or less than -9223372036854775808
- the decimal constant for a BI field was greater than 18446744073709551615 or less than +0
- the number of digits (including leading zeros) in the decimal constant for an FI or BI field was greater than 31
- the length for a Y2 field was not 2 for Y2C, Y2Z, Y2P or Y2S, or 1 for Y2D or Y2B, or 3-6 for Y2T or Y2W, or 2-3 for Y2U or Y2X, or 3-4 for Y2V or Y2Y
- a Y2 field was compared to another Y2 field with a different number of non-year digits
- a Y2 field was compared to a Y constant with a different number of non-year digits
- a Y2 field other than Y2S, Y2T or Y2W was compared to Y'LOW', Y'BLANKS' or Y'HIGH'
- a Y2 field was compared to a constant that was not a Y constant

m indicates the number of the relational condition in which the error was found (starting at 1 for the first relational condition). For example, in

```
INCLUDE COND=(5,2,CH,EQ,8,2,CH,OR,
   11,257,BI,EQ,301,257,BI)
```

the second relational condition (after the OR) has the error described in the first bullet above, so m is 2.

System Action: The program terminates.

Programmer Response: Correct the field length or constant in error in relational condition m.

ICE222A

ICE222A n BYTE FIXED RECORD LENGTH IS NOT EQUAL TO m BYTE LRECL FOR ddname

Explanation: Critical. The LRECL specified or retrieved for the fixed-length OUTFIL data set was not equal to the computed length of the output records for that data set. You cannot use the LRECL value to pad the OUTFIL records or to truncate the records produced by OUTREC, BUILD, OVERLAY, IFTHEN BUILD or IFTHEN OVERLAY operand processing. The values shown in the message are as follows:

- n is the computed length of the output records for the OUTFIL group
- m is the specified or retrieved LRECL of the OUTFIL data set
- ddname indicates the OUTFIL data set for which padding or truncation was required

System Action: The program terminates.

Programmer Response: Take one of these actions as appropriate:

- Do not set the LRECL explicitly. Instead, let DFSORT set the LRECL to the computed record length.
- If you are using IFTHEN operands, specify IFOUTLEN=m. (Remember to allow an extra byte for OUTFIL report data sets for the ANSI carriage control character unless you specify the REMOVECC operand.)
- If you are not using IFTHEN operands, ensure that the computed length for the BUILD or OVERLAY operand is equal to m. (Remember to allow an extra byte for OUTFIL report data sets for the ANSI carriage control character unless you specify the REMOVECC operand.)

ICE223A

ICE223A REPORT FIELD ERROR

Explanation: Critical. The LINES, HEADER1, TRAILER1, HEADER2, TRAILER2, or SECTIONS parameter of an OUTFIL statement contained an invalid column, report element, position, length, format, keyword, pattern, sign, or constant. Some common errors are :

- A 0 value was used.
- A null value was used where it was not permitted.
- A null string, pattern, or sign was used.
- A column was greater than 32752, preceded / or n/ (new line), or was followed by another column.
- A column overlapped the previous output field in the report record (a missing new line (/ or n/) to end the current report record and start the next one can cause this error).
- A position plus length was greater than 32753.
- DATE=(abcd) or DATENS(abc) was specified with a, b or c not M, D, Y or 4, with M, D, Y or 4 specified more than once, or with Y and 4 both specified.

- YDDD=(abc) or YDDDNS=(ab) was specified with a or b not D, Y or 4, with D, Y or 4 specified more than once, or with Y and 4 both specified.
- TIME=(abc) or TIMENS=(ab) was specified with ab not 12 or 24.
- The length for an input field or section break field was greater than 256 bytes.
- A repetition factor was greater than 4095 for a blank, character string or hexadecimal string report element, or greater than 255 for a blank lines report element or a section skip line count.
- A character or hexadecimal constant was longer than 256 bytes.
- An invalid digit or an odd number of digits was specified for a hexadecimal string.
- The length for a statistics field was greater than 8 bytes for BI or FI, 16 bytes for PD, 31 bytes for ZD, 32 bytes for CSF/FS, or 44 bytes for UFF or SFF.
- More than 31 digits or 44 characters were specified in an edit pattern.
- SIGNz (where z is not S) was specified with Mn or without EDIT or EDxy.
- x, y, or z in EDxy or SIGNz were the same character.
- The value for LENGTH was greater than 44.
- The value for LINES was greater than 255.
- A section break field was not followed by SKIP, HEADER3, or TRAILER3.
- A statistics field was specified in HEADER1, HEADER2, or HEADER3.
- HEADER3, TRAILER3, SKIP, or PAGEHEAD was specified more than once after a section break field.

System Action: The program terminates.

Programmer Response: Correct the invalid value.

ICE230A

ICE230A n BYTE HEADER/TRAILER RECORD LENGTH EXCEEDS m BYTE LRECL FOR ddname

Explanation: Critical. The specified, retrieved, or computed LRECL for the OUTFIL data set was less than the computed length for the report records for that data set. The values shown in the message are as follows:

- n is the computed length of the report records for the OUTFIL group
- m is the specified, retrieved, or computed LRECL of the OUTFIL data set
- ddname indicated the OUTFIL data set for the report

System Action: The program terminates.

Programmer Response: Use the BUILD, OVERLAY, IFTHEN or IFOUTLEN operand to force a length for the data records that is longer than any report record, and either let DFSORT compute and set the LRECL, or ensure that the computed LRECL is equal to the existing or specified LRECL. (Remember to allow an extra byte in the LRECL for the ANSI carriage control character unless you specify the REMOVECC parameter.)

For example, if your data records are 40 bytes, but your longest report record is 60 bytes, you could use an OVERLAY parameter such as:

```
OUTFIL HEADER1=(51:DATE=(4MD/)),OVERLAY=(80:X)
```

DFSORT will then set the LRECL to 81 (1 byte for the ANSI carriage control character plus 80 bytes for the length of the data records), and pad the data records with blanks on the right.

ICE241A

ICE241A {*INREC|*OUTREC|ddname} IFTHEN n OVERLAY COLUMN OVERLAPS RECORD DESCRIPTOR WORD

Explanation: Critical. For variable-length record processing, the OVERLAY or IFTHEN OVERLAY operand of an INREC, OUTREC or OUTFIL statement specified an overlay item that overlapped the record descriptor word (RDW). Only data bytes, which start at position 5 for variable-length records, can be overlaid.

The specific cause of the error is indentified as follows:

- *INREC and n=0 indicates that the OVERLAY operand of the INREC statement caused the error.
- *OUTREC and n=0 indicates that the OVERLAY operand of the OUTREC statement caused the error.
- ddname and n=0 indicates that the OVERLAY operand of an OUTFIL statement caused the error. ddname identifies the first data set in the associated OUTFIL group.
- *INREC and n>0 indicates that an IFTHEN OVERLAY operand of the INREC statement caused the error. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the INREC statement).
- *OUTREC and n>0 indicates that an IFTHEN OVERLAY operand of the OUTREC statement caused the error. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the OUTREC statement).
- ddname and n>0 indicates that an IFTHEN OVERLAY operand of an OUTFIL statement caused the error. ddname identifies the first data set in the associated OUTFIL group. n identifies the number of the associated IFTHEN clause (starting at 1 for the first IFTHEN clause in the OUTFIL statement).

The error is one of the following:

- c: was not specified for the first OVERLAY item so the default of 1: was used for that item.
  Example:

  ```
  OVERLAY=(C'ABC')
  ```

- c: was specified for an overlay item with a value for c which was less than 5.
  Example:

  ```
  OVERLAY=(3:C'ABC')
  ```

System Action: The program terminates.

Programmer Response: Specify c: with a value of 5 or more for the first OVERLAY item. Ensure that c is 5 or more for any other c: values you specify.
Example:

```
OVERLAY=(8:C'ABC',1,2,HEX,25:5C'*')
```

ICE273A

ICE273A SYMBOL OR VALUE IS TOO LONG

Explanation: Critical. The SYMNAMES statement has one of the following errors:

- The symbol is longer than 50 characters.

- The value contains a number longer than 31 significant digits.
- The constant contains a string longer than 64 characters.

System Action: The program terminates.

Programmer Response: Specify a symbol, value or string that is less than or equal to the allowed limit.

ICE276A

ICE276A RESERVED WORD - NOT ALLOWED FOR SYMBOL

Explanation: Critical. The SYMNAMES statement specifies a DFSORT/ICETOOL reserved word for the symbol. Reserved words cannot be used for symbols. The reserved words are as follows (uppercase only as shown): A, AC, ADD, ALL, AND, AQ, ASL, AST, BI, CH, CLO, COPY, COUNT, COUNT15, CSF, CSL, CST, CTO, D, DATE, DATE1, DATE1P, DATE2, DATE2P, DATE3, DATE3P, DATE4, DC1, DC2, DC3, DE1, DE2, DE3, DIV, DT1, DT2, DT3, D1, D2, E, F, FI, FL, FS, H, HEX, LS, MAX, MIN, MOD, MUL, Mn, Mnn, NONE, NUM, OL, OR, OT, PAGE, PAGEHEAD, PD, PD0, SEQNUM, SFF, SS, SUB, SUBCOUNT, SUBCOUNT15, TC1, TC2, TC3, TC4, TE1, TE2, TE3, TE4, TIME, TIME1, TIME1P, TIME2, TIME2P, TIME3, TIME3P, TM1, TM2, TM3, TM4, TS, UFF, VALCNT, VLEN, X, Y2x, Y2xx, Z, ZD, ZDF and ZDC where n is 0-9 and x is any character.

System Action: The program terminates.

Programmer Response: Use a symbol that is not one of the reserved words, such as a lowercase or mixed case version of the word being used. For example, you could use Valcnt which is not a reserved word instead of VALCNT which is.

ICE604A

ICE604A ERROR IN KEYWORD, PARAMETER, OR DELIMITER

Explanation: Critical. The statement contained an error in an operand (keyword, parameter) or a delimiter was incorrect or missing. Some common errors are:

- A keyword or parameter was misspelled. Example: ALLDUP instead of ALLDUPS
- A keyword was used with an operator for which it is not valid. Example: NOSIGN was used with an operator other than VERIFY.
- A parameter or value was used with a keyword for which it is not valid. Example: VSAMTYPE(U) instead of VSAMTYPE(F) or VSAMTYPE(V).
- The cccc value for USING(cccc) was SYSc or was not four characters. Example: USING(ABC) instead of USING(ABCD) or USING(SYS1) instead of USING(SYX1).
- A left or right parenthesis was missing. Example: FROM IN instead of FROM(IN)
- A blank was used inside a parentheses. Example: FROM(IN) instead of FROM(IN)
- A continuation indicator (-) was used incorrectly. Example: TO(OUT1,- instead of TO(OUT1,OUT2) -
- Parameters were not separated by a comma or semicolon. Example: ON(3:5:ZD) instead of ON(3,5,ZD) or ON(3;5;ZD)
- A numeric value was specified incorrectly. Example: ON(0,3,ZD) instead of ON(1,3,ZD) or LIMIT(+1) instead of LIMIT(1)

- An operand that can be specified only once per operator was specified more than once. Example: COPY TO(OUT1) TO(OUT2) instead of COPY TO(OUT1,OUT2)
- Mutually exclusive operands were used. Example: BLANK and PLUS for DISPLAY or WITHEACH and VLENOVLY for SPLICE.
- A numeric value was too low or too high. Example: LINES(9) or LINES(1000)
- A string was not enclosed in apostrophes. Example: HEADER(Revenue), HEADER('Revenue), HEADER('Revenue''), or HEADER(''Revenue'')instead of HEADER('Revenue')
- A string exceeded the character limit allowed. Example:
  TITLE('This string is longer
  than the limit of 50 characters
  for TITLE')

  Note: In your ICETOOL statement, the entire operand must be on one line.
- A parameter was specified incorrectly. Example: DATE(DMY) instead of DATE(DMY.)
- NOHEADER was used when HEADER(NONE) was required. Example: HEADER('Name') NOHEADER instead of HEADER('Name') HEADER(NONE)
- Mutually exclusive items were specified within an operand. Example: DATE(YM4/) instead of DATE(YMD/) or DATE(DM4/)
- A formatting item was specified for an operator other than DISPLAY or OCCUR. Example: SELECT ON(1,5,ZD,A1) instead of SELECT ON(1,5,ZD)
- A symbol was specified where it is not allowed. Example: LINES(Max_Lines) instead of LINES(50)
- Too many 9's were specified in the pattern for E'pattern'. Example:
  ON(VALCNT,E'999-999-999-999-999-9') instead of
  ON(VALCNT,E'99-999-999-999-999-9')
- Too many characters were specified in the pattern for E'pattern'. Example:
  ON(VALCNT,E'*999**999**999**999**999*') instead of
  ON(VALCNT,E'999**999**999**999**999*')
- A required parameter was missing. Example: ON(25,5) instead of ON(25,5,ZD)
- A parameter was specified where it isn't allowed. Example: WITH(25,5,ZD) instead of WITH(25,5)

System Action: This operation is terminated.

Programmer Response: A $ marks the point at which the error was detected. Correct the error.

ICE608I

ICE608I MINIMUM: snnnnnnnnnnnnnnn, MAXIMUM: snnnnnnnnnnnnnnn

Explanation: Indicates the minimum and maximum for the field indicated in the ICE607I message preceding this message. Each value consists of a + or - sign and 15 decimal digits (padded with zeros on the left as needed). If the values could not be determined due to an error (as indicated in a previous message), asterisks were printed for the values.

Note: ICE648I messages are printed for the minimum and maximum values (instead of ICE608I) if either value contains more than 15 significant digits, or if LMSG is specified.

System Action: None.

Programmer Response: None unless asterisks were printed for the values, in which case you should correct the error indicated by the previous error message for this operation.

If you want all of the minimum and maximum values displayed with 31 digits, specify the LMSG keyword for this STATS operator.

ICE609I

ICE609I AVERAGE: snnnnnnnnnnnnnnn, TOTAL: snnnnnnnnnnnnnnn

Explanation: Indicates the average and total for the field indicated in the ICE607I message preceding this message. Each value consists of a + or - sign and 15 decimal digits (padded with zeros on the left as needed). If a value could not be determined due to an error (as indicated in a previous message), asterisks were printed for the value.

Note: ICE648I messages are printed for the average and total values (instead of ICE609I) if either value contains more than 15 significant digits, or if LMSG is specified.

System Action: None.

Programmer Response: None unless asterisks were printed for a value, in which case you should correct the error indicated by the previous error message for this operation.

If you want all of the average and total values displayed with 31 digits, specify the LMSG keyword for this STATS operator.

ICE611A

ICE611A TOTAL FOR {(p,m,f)|(VLEN)} OVERFLOWED n DECIMAL DIGITS

Explanation: Critical. The total for the indicated ON(p,m,f) or ON(VLEN) field (STATS or DISPLAY) exceeded the number of digits (n) allowed for it. n is 31 for STATS, or 15 or 31 for DISPLAY depending on the number of digits allowed for the total.

System Action: This operation is terminated.

For a STATS operator, asterisks are printed in ICE648I messages for the average and total for this field.

For a DISPLAY operator, asterisks are printed for this field in any BAVERAGE, BTOTAL, AVERAGE, or TOTAL lines requested.

Programmer Response:

If n is 15, specify an Ndd or Udd formatting item large enough to prevent overflow of the total (that is, use an appropriate Ndd or Udd value between 16 and 31).

If n is 31 and you need the average or total for this field, use the STATS or DISPLAY operator for subsets of the data set which do not cause the total to overflow. Use the statistics for the subsets to determine the needed statistics for the field.

ICE618A

ICE618A INVALID (p,m,f) VALUE - RECORD: nnnnnnnnnnnnnnn

Explanation: Critical. ICE618A and ICE649A identify an invalid decimal value in a specified field. One of the following was found:

- An invalid digit (A-F) in a field specified for a DISPLAY, OCCUR, RANGE, STATS, or VERIFY operator.
- An invalid sign (0-9) in a field specified for a VERIFY operator (NOSIGN was not specified).

(p,m,f) is a field you specified for this operator.

The invalid value was found in record number nnnnnnnnnnnnnnn (prints OCCUR operator, nnnnnnnnnnnnnnn is the sorted record number, and thus may not be useful.

h...h in the ICE649A message that follows this ICE618A message is the invalid value in hexadecimal.

System Action:

- For a DISPLAY operator with an invalid BREAK value, this operation is terminated.
- For a DISPLAY operator with an invalid ON value, asterisks are printed for this value in the data line and in any statistics lines requested. If this incorrect value causes the limit for decimal values to be reached, this operation is terminated after the current record is printed in the list data set. Otherwise, processing continues.
- For an OCCUR or RANGE operator, this operation is terminated.
- For a STATS operator, asterisks were printed in messages ICE608I, ICE609I or ICE648I for minimum, maximum, average and total for this field.
- For a VERIFY operator, if this bad value caused the limit for invalid decimal values to be reached, this operation is terminated. Otherwise, processing continues.

Programmer Response: Correct the invalid digit or sign in the identified field. The VERIFY or DISPLAY operator can be used to print all the invalid values and their relative record numbers.

ICE634A

ICE634A VALUE FOR (p,m,f) EXCEEDS 31 DIGITS

Explanation: Critical. A value for the indicated FS or CSF field had 32 digits, or a value for the indicated UFF or SFF field had 32 or more digits, exceeding the limit of 31 digits.

System Action: This operation is terminated.

Programmer Response: Limit all FS, CSF, UFF and SFF data values for this operation to 31 digits. The DISPLAY operator with CH format can be used to visually identify the values with 32 digits.

ICE640A

ICE640A INVALID FORMATTING ITEM

Explanation: Critical. An ON(p,m,f,formatting), ON(VLEN,formatting) or ON(NUM,formatting) operand for this DISPLAY operator, or an ON(p,m,f,formatting), ON(VLEN,formatting) or ON(VALCNT,formatting) operand for this OCCUR operator, contained an invalid formatting item as follows:

- The formatting item was not /x (/x can be /D, /C, /K, /DK, /CK, /M, /G, /KB, /MB, or /GB), L'string', F'string', T'string', E'pattern', NOST, LZ, Ndd, Udd or a valid mask (mask can be A0-A5, B1-B6, C1-C6, D1-D6, E1-E4, F1-F5, or G1-G6).
- /x, F'string', E'pattern', LZ, NOST, Ndd, Udd or a mask was specified for a character field.
- /x or NOST was specified for ON(NUM,formatting) or for OCCUR.
- /x, NOST or Ndd was specified for BREAK(p,m,f,formatting).
- More than one /x was specified, Ndd and Udd were both specified, more than one mask was specified, or L'string', F'string', T'string', E'pattern', Udd, or Ndd was specified more than once.
- L'', F'', T'' or E'' was specified.
- F'string' or a mask was specified with E'pattern'.
- dd for Ndd or Udd was not two digits from 01 to 31.
- dd for Ndd or Udd was greater than 15 for ON(NUM,formatting) or ON(VALCNT,formatting).

System Action: This operation is terminated.

Programmer Response: A $ marks the point at which the error was detected. Correct the error.

ICE644A

ICE644A {VALUE|TOTAL} FOR {(p,m,f)|(VLEN)} OVERFLOWED n DECIMAL DIGITS

Explanation: Critical. A value for the indicated ON(p,m,f) or ON(VLEN) field (DISPLAY or OCCUR) or BREAK(p,m,f) field (DISPLAY), or the total for the indicated ON(p,m,f) field, exceeded the number of digits (n) you allowed for it. n is the number of digits allowed for the value or total as determined from the length and format of the field, or the Ndd or Udd formatting item you specified.

System Action:

- For a DISPLAY operator with an overflowing BREAK value, this operation is terminated.
- For a DISPLAY operator with an overflowing ON value, asterisks are printed for this value in the data line and in any statistics lines requested. Processing continues.
- For a DISPLAY operator with an overflowing total, asterisks are printed for the total. Processing continues
- For an OCCUR operator with an overflowing ON value, this operation is terminated.

Programmer Response: Specify an Ndd or Udd formatting item large enough to prevent overflow of the value or total (that is, use an appropriate Ndd or Udd value between n+1 and 31).

ICE645A

ICE645A {(NUM)|(VALCNT)} OVERFLOWED n DECIMAL DIGITS

Explanation: Critical. The number of digits (n) allowed from the Ndd or Udd formatting item you specified was too small, as follows:

- (NUM) indicates a record number for this DISPLAY operator exceeded the number of digits you allowed for it.
- (VALCNT) indicates a value count for this OCCUR operator exceeded the number of digits you allowed for it.

System Action: This operation is terminated.

Programmer Response: Specify an Ndd or Udd formatting item large enough to prevent overflow of the record number or value count (that is, use an appropriate Ndd or Udd value between n+1 and 15).

ICE648I

ICE648I statistic: snnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn

Explanation: Indicates the minimum, maximum, average or total for the field indicated in the ICE607I message preceding this message. statistic is MINIMUM, MAXIMUM, AVERAGE or TOTAL. Each value consists of a + or - sign and 31 decimal digits (padded with zeros on the left as needed).

Note: An ICE608I message is printed for the minimum and maximum values (instead of ICE648I messages) if both values contains less than 16 significant digits, unless LMSG is specified. An ICE609I message is printed for the average and total values (instead of ICE648I messages) if both values contain less than 16 significant digits, unless LMSG is specified.

System Action: None.

Programmer Response: None unless asterisks were printed for the AVERAGE and TOTAL values, in which case you should correct the error indicated by the previous error message for this operation.

If you want all of the minimum, maximum, average and total values displayed with 31 digits, specify the LMSG keyword for this STATS operator.

ICE649A

ICE649A HEX VALUE: h...h

Explanation: Critical. Displays the invalid value associated with preceding message ICE618A. h...h is the invalid value in hexadecimal. See the **Explanation** of message ICE618A for details.

System Action: See the **System Action** of message ICE618A for details.

Programmer Response: See the **Programmer Response** of message ICE618A for details.

ICE651I

ICE651I RECORD COUNT MEETS CRITERIA - RC=4 SET

Explanation: RC4 and EMPTY, NOTEMPTY, HIGHER(n), LOWER(n), EQUAL(n), or NOTEQUAL(n) was specified for this COUNT operator. Since the record count met the specified criteria, RC=4 was set for this COUNT operation.

System Action: Processing continues and ICETOOL passes back a return code of 4 unless it sets a higher return code for some other reason.

Programmer Response: None.