User Guide for DFSORT PTFs UK51706 and UK51707

November, 2009

Frank L. Yaeger

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

DFSORT 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 **UK51706** and z/OS DFSORT V1R10 PTF **UK51707**, which were first made available in **November, 2009**.

These PTFs provide important enhancements to DFSORT and DFSORT's ICETOOL for various types of two file join applications (JOINKEYS, JOIN, REFORMAT, JKFROM); date field conversions (Y2x, Y4x, TOJUL, TOGREG, WEEKDAY, DT, DTNS); merge operations (MERGE operator); alternate ddnames for merge files (MERGEIN); 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 UK51706 and UK51707

Introduction

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

DFSORT, together with DFSMS and RACF, form the strategic product base for the evolving system-managed storage environment. DFSMS 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 **UK51706** and z/OS DFSORT V1R10 PTF **UK51707**, which were first made available in **November, 2009**, provide important enhancements to DFSORT and DFSORT's ICETOOL for various types of two file join applications (JOINKEYS, JOIN, REFORMAT, JKFROM); date field conversions (Y2x, Y4x, TOJUL, TOGREG, WEEKDAY, DT, DTNS); merge operations (MERGE operator); alternate ddnames for merge files (MERGEIN); 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.

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 V1R5 PTF UK51706 or z/OS DFSORT V1R10 PTF UK51707. The information in this paper will be included in the z/OS DFSORT books at a later date.

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 Changes

JOINKEYS Application

A new JOINKEYS application helps you to perform various "join" operations on two files by one or more keys. A JOINKEYS application makes it easy to create joined records in a variety of ways including inner join, full outer join, left outer join, right outer join, and unpaired combinations. The two input files can be of different types (fixed, variable, VSAM, and so on) and have keys in different locations.

Three new control statements can be used for a JOINKEYS application. One JOINKEYS statement is required for each input file to indicate the ddname of the file, describe the keys, indicate whether the file is already sorted by those keys, and so on. An inner join is performed by default, but a JOIN statement can be used to specify a different type of join. A REFORMAT statement is used to describe the fields from the two files to be included in the joined records, and optionally an indicator of where the key was found ('B' for both files, '1' for file1 only or '2' for file2 only).

The records from the two input files can be processed in a variety of ways before and after they are joined.

A new JKFROM operand makes it easy to use a JOINKEYS application with a COPY or SORT operator of ICETOOL.

DFSORT symbols can be used for fields and constants specified in the JOINKEYS and REFORMAT statements and in the other DFSORT statements used with a JOINKEYS application.

Date Field Conversions

Y2x, Y4x, TOJUL, TOGREG, WEEKDAY, DT and DTNS keywords for the BUILD and OVERLAY operands make it easy to convert input date fields of one type to corresponding output date fields of another type, and to convert a date field to a corresponding day of the week in several forms.

The date field can be in either Julian or Gregorian form, with 2-digit or 4-digit years, in CH, ZD and PD format. For example, C'ccyyddd', P'yymmdd', Z'dddccyy', C'ccyymmdd', P'dddyy', and so on.

CH output date fields can also be displayed with separators. For example, C'ccyy/ddd', C'mm-dd-ccyy', and so on.

The day of the week can be displayed as a 1 digit, 3 character or 9 character constant. For example, C'4', C'WED' or C'WEDNESDAY' for Wednesday.

DFSORT symbols can be used for fields in date conversions.

MERGE Operator

MERGE is a new operator of ICETOOL that allows you to merge up to 50 input files that are already in order by the same keys. MERGE makes it easy to include DFSORT MERGE operations in your ICETOOL jobs.

Various options of MERGE allow you to define the ddnames for the input data sets (FROM), the ddnames for the output data sets (TO), the MERGE and other DFSORT control statements to be used for the MERGE operation (USING), and so on.

DFSORT symbols can be used in the DFSORT statements used with a MERGE operator.

MERGEIN Alternate ddnames

A new MERGEIN operand can be specified on an OPTION statement in DFSPARM or an extended parameter list to supply alternate ddnames for input data sets used for a MERGE application. MERGEIN makes it easy to use any ddnames you like instead of SORTINnn ddnames. You can specify up to 100 ddnames to be merged with MERGEIN.

New 24-Bit Parameter List Codes

DFSORT now accepts and processes the following as 24-Bit parameter list control statement entry codes: X'0E', X'0F' and X'11'.

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:

- New reserved word for symbols

  Y4x, where Y is uppercase and x is any character, is a new DFSORT/ICETOOL reserved word which is no longer allowed as a symbol. If you used Y4x as a symbol, you must change it. For example, if you used Y4T, you can change it to y4t.

- MERGE ddname in message ICE217A

  When message ICE217A is issued for a MERGE application, DFSORT will now display the actual ddname for the MERGE input data set rather than 'SORTINNN'.

  Any automated actions based on the ICE217A message should be evaluated since the ddname will change for a MERGE application.

JOINKEYS Application for Joining Two Files

Introduction

You can perform various types of "join" applications on two files (F1 and F2) by one or more keys with DFSORT using the following statements:

- JOINKEYS: You must specify two JOINKEYS statements; one for the F1 file and another for the F2 file. A separate subtask will be used to process each file and a main task will be used to process the joined records from the two files.

  Each JOINKEYS statement must specify the ddname of the file it applies to and the starting position, length and sequence of the keys in that file. You can also optionally specify if the file is already sorted by the keys and if sequence checking of the keys is not needed; if the file has fixed-length or variable-length records; to stop reading the file after n records; a 2-byte id to be used for the message and control data set for the subtask used to process the file, and if a subset of the records is to be processed based on a logical expression.

- JOIN: If you don't specify a JOIN statement, only paired records from F1 and F2 are kept and processed by the main task as the joined records (inner join). You can optionally specify a JOIN statement to have the main task keep and process: unpaired F1 records as well as paired records (left outer join); unpaired F2 records as well as paired records (right outer join); unpaired F1 and F2 records as well as paired records (full outer join); only unpaired F1 records; only unpaired F2 records, or only unpaired F1 and F2 records.

- REFORMAT: You would normally specify a REFORMAT statement to indicate the F1 and/or F2 fields you want in the joined records. You can optionally specify an indicator of where the key was found, and a FILL character to be used for missing bytes. If a JOIN statement with ONLY is specified, the REFORMAT statement is optional.

F1 and F2 can be any type of sequential or VSAM file supported by DFSORT for SORTIN and can have different attributes (for example, F1 can have RECFM=FB and LRECL=100 and F2 can have RECFM=VB and LRECL=254, or F1 can be a VSAM ESDS and F2 can be a PDS member).

F1 will be sorted or copied by "subtask1". An E35 exit will be used to pass the needed fields from the F1 records to the "main task" (an intermediate output data set is not used or required). A subset of the DFSORT statements, such as INCLUDE, OMIT, INREC, SUM and OPTION, will be available for processing the F1 records.

F2 will be sorted or copied by "subtask2". An E35 exit will be used to pass the needed fields from the F2 records to the "main task" (an intermediate output data set is not used or required). A subset of the DFSORT statements, such as INCLUDE, OMIT, INREC, SUM and OPTION, will be available for processing the F2 records.

The "main task" will use an E15 to join the records passed from the E35 of subtask1 and E35 of subtask2. The joined records will be processed as the input records for a sort or copy application. Most of the control statements and options available for a DFSORT sort or copy application will be available for processing the joined records.

Note: Since a JOINKEYS application uses three tasks, it can require more storage than a regular DFSORT application. You may need to use REGION=0M for some JOINKEYS applications.

JOINKEYS Application Processing Diagram

Here's a pictorial representation of the processing performed for a JOINKEYS Application, and the order in which the various functions are performed.

```
+------------------------+          +------------------------+
| SORTJNF1 (F1 file)     |          | SORTJNF2 (F2 file)     |
+------------------------+          +------------------------+
           |                                   |
           V                                   V
+------------------------+          +------------------------+
| Subtask1 for F1        |          | Subtask2 for F2        |
|  Messages: JNF1JMSG    |          |  Messages: JNF2JMSG    |
|  Control:  JNF1CNTL    |          |  Control:  JNF2CNTL    |
--------------------------         --------------------------
| <SKIPREC>              |          | <SKIPREC>              |
| <E15>                  |          | <E15>                  |
| <INCLUDE/OMIT>         |          | <INCLUDE/OMIT>         |
| <STOPAFT>              |          | <STOPAFT>              |
| <INREC>                |          | <INREC>                |
| COPY or SORT (1)       |          | COPY or SORT (2)       |
| <SUM>                  |          | <SUM>                  |
--------------------------         --------------------------
| E35 passes F1 fields   |          | E35 passes F2 fields   |
| to main task's E15 (3) |          | to main task's E15 (4) |
+------------------------+          +------------------------+
          |                                       |
          +--------------------+------------------+
                               |
                               V
          +---------------------------------------+
          | Main task for joined records          |
          |  Messages:  SYSOUT                    |
          |  Control:   SYSIN, DFSPARM,           |
          |             SORTCNTL, parmlist        |
          -----------------------------------------
          | E15 joins fields passed by            |
          | subtask1's E35 and subtask2's E35     |
          | as directed by JOINKEYS, JOIN and     |
          | REFORMAT statements (5)               |
          -----------------------------------------
          | <INCLUDE/OMIT>                        |
          | <STOPAFT>                             |
          | <INREC>                               |
          | <SORT or COPY - one required>         |
          | <SUM>                                 |
          | <OUTREC>                              |
          | <E35/SORTOUT/OUTFIL - one required>   |
          +---------------------------------------+
Legend:
(1) COPY is used automatically for subtask1 if the SORTED operand is
    specified in the F1 JOINKEYS statement.  Otherwise, SORT is used
    automatically with the binary keys specified in the FIELDS operand
    of the F1 JOINKEYS statement.
(2) COPY is used automatically for subtask2 if the SORTED operand is
    specified in the F2 JOINKEYS statement.  Otherwise, SORT is used
    automatically with the binary keys specified in the FIELDS operand
    of the F2 JOINKEYS statement.
(3) An E35 exit is used for subtask1 automatically.  (An intermediate
    output file is not required or used.)
(4) An E35 exit is used for subtask2 automatically.  (An intermediate
    output file is not required or used.)
(5) An E15 exit is used for the main task automatically. (Intermediate
    input files are not required or used.)
```

For direct invocation of DFSORT (e.g. PGM=SORT), the JOINKEYS, JOIN and REFORMAT statements can be specified in SYSIN or DFSPARM. For program invocation of DFSORT (e.g. LINK EP=SORT), the JOINKEYS, JOIN and REFORMAT statements can be specified in the caller's parameter list, in SORTCNTL or in DFSPARM.

Subtask1 reads the F1 file. It uses COPY or SORT, and an E35 exit (with no output data set) automatically to pass the needed F1 fields to the main task's E15 exit. Subtask1 can optionally use the other listed control statements from JNF1CNTL. Subtask1 messages are displayed in JNF1JMSG.

Subtask2 reads the F2 file. It uses COPY or SORT, and an E35 exit (with no output data set) automatically to pass the needed F2 fields to the main task's E15 exit. Subtask2 can optionally use the other listed control statements from JNF2CNTL. Subtask2 messages are displayed in JNF2JMSG.

The main task uses an E15 exit automatically. The E15 creates the joined records by accessing the F1 fields passed by subtask1's E35 and the F2 fields passed by subtask2's E35. The main task writes the output to SORTOUT and/or OUTFIL or uses a supplied E35 exit to "delete" all of the records. For direct invocation of DFSORT, the main task can optionally use the other listed control statements from SYSIN or DFSPARM. For program invocation of DFSORT, the main task can optionally use the other listed control statements from the caller's parameter list, in SORTCNTL or in DFSPARM. The main task's messages are displayed in SYSOUT.

The starting position in the fields you specify for the subtasks or main task must reflect any reformatting of the records you do at each stage. This includes using the starting positions of the joined records for main task functions.

Examples:

- If you specify INREC and SUM statements in JNF1CNTL for subtask1, the FIELDS operands in the JOINKEYS, REFORMAT, and SUM statements must specify the starting positions in the F1 records as reformatted by INREC.
- If you specify an INCLUDE statement in SYSIN for the main task, the COND operand must specify the starting positions in the joined records as reformatted by REFORMAT.

Sample JOINKEYS Applications

Here's the JCL and control statements for a simple JOINKEYS application to do an inner join (also known as a cartesian join) which joins all paired records.

```
//S1   EXEC  PGM=SORT
//SYSOUT DD SYSOUT=*
//SORTJNF1 DD DSN=INPUT1,DISP=SHR
//SORTJNF2 DD DSN=INPUT2,DISP=SHR
//SORTOUT DD SYSOUT=*
//SYSIN DD *
* Control statements for JOINKEYS application
  JOINKEYS FILE=F1,FIELDS=(15,2,A,7,4,A)
  JOINKEYS FILE=F2,FIELDS=(21,2,A,23,4,A)
  REFORMAT FIELDS=(F2:1,70,F1:1,60)
* Control statements for main task (joined records)
  SORT FIELDS=COPY
/*
```

Here's the JCL and control statements for a JOINKEYS application which omits certain records and normalizes keys for F1 and F2, and retains only sorted, non-duplicate, unpaired records from F1. The F1 and F2 files are already in order by the specified JOINKEYS FIELDS. DFSORT symbols are used for various fields and constants.

```
//S2   EXEC  PGM=SORT
//SYSOUT DD SYSOUT=*
//SYMNAMES DD *
IN1_dept,11,3,ch
IN1_target,'J82'
IN1_normal_key,27,5,ZD
IN1_output,1,60
IN2_dept,14,3,ch
IN2_target,'M25'
IN2_PD_key,21,3,PD
IN2_normal_key,=,5,ZD
join_sort_key,45,8,UFF
//FIX DD DSN=F.INPUT,DISP=SHR
//VAR DD DSN=V.INPUT,DISP=SHR
//SORTOUT DD DSN=FIXOUT1,DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,(5,5)),UNIT=SYSDA
//JNF1CNTL DD *
* Control statements for subtask1 (F1)
  OMIT COND=(IN1_dept,EQ,IN1_target)
  INREC OVERLAY=(IN1_normal_key:IN1_normal_key,TO=ZDF,LENGTH=5)
//JNF2CNTL DD *
* Control statements for subtask2 (F2)
  OMIT COND=(IN2_dept,EQ,IN2_target)
  INREC OVERLAY=(IN2_PD_key:IN2_PD_key,TO=ZDF,LENGTH=5)
//SYSIN DD *
* Control statements for JOINKEYS application
  JOINKEYS F1=FIX,FIELDS=(IN1_normal_key,D),SORTED
  JOINKEYS F2=VAR,FIELDS=(IN2_normal_key,D),SORTED
  JOIN UNPAIRED,F1,ONLY
  REFORMAT FIELDS=(F1:IN1_output)
* Control statements for main task (joined records)
  SORT FIELDS=(join_sort_key,A)
  SUM FIELDS=NONE
/*
```

JCL for JOINKEYS Application

The required JCL statements used for a JOINKEYS application are as follows. See *z/OS DFSORT Application Programming Guide* for general information on DFSORT message, control, input, output, work and symbols data sets.

- //stepname EXEC PGM=SORT

  Invokes DFSORT. PGM=ICEMAN can also be used.

  Note: Since a JOINKEYS application uses three tasks, it can require more storage than a regular DFSORT application. You may need to use REGION=0M for some JOINKEYS applications.

- //SYSOUT DD

  Messages from the main task. An alternate ddname can be supplied with MSGDDN=ddname in //DFSPARM.

- //SORTJNF1 DD

  F1 input file (read by subtask1). The ddname is SORTJNF1 (or ccccJNF1 if SORTDD=cccc is in effect) if FILE=F1 or FILES=F1 is specified on the JOINKEYS statement. An alternate ddname can be supplied with F1=ddname on the JOINKEYS statement.

  Note: The F1 data set is treated as a SORTIN data set by subtask1 and is subject to the rules for a SORTIN data set documented in *z/OS DFSORT Application Programming Guide*.

- //SORTJNF2 DD

  F2 input file (read by subtask2). The ddname is SORTJNF2 (or ccccJNF2 if SORTDD=cccc is in effect) if FILE=F2 or FILES=F2 is specified on the JOINKEYS statement. An alternate ddname can be supplied with F2=ddname on the JOINKEYS statement.

  Note: The F2 data set is treated as a SORTIN data set by subtask2 and is subject to the rules for a SORTIN data set documented in *z/OS DFSORT Application Programming Guide*.

- //SORTOUT DD or //outfil DD

  Output of joined records (written by main task). An alternate ddname can be supplied with SORTOUT=ddname or SORTDD=cccc in //DFSPARM or with the OUTFIL statement in //SYSIN or //DFSPARM.

- //DFSPARM DD, //SYSIN DD or //SORTCNTL DD

  Control statements for the main task including JOINKEYS, JOIN, REFORMAT, OPTION, SORT, INCLUDE or OMIT, SUM, OUTREC, RECORD, ALTSEQ, MODS and OUTFIL. If SKIPREC or E15 is specified, it will not be used.

  DFSPARM can be used if DFSORT is invoked directly or called from a program. SYSIN can be used if DFSORT is invoked directly. SORTCNTL can be used if DFSORT is called from a program; an alternate ddname of ccccCNTL can be supplied with SORTDD=cccc in DFSPARM.

  Note: If DFSORT is called from a program, the control statements for the main task can be supplied using the parameter list.

The optional JCL statements used for a JOINKEYS application are as follows:

- SYMNAMES DD

  DFSORT symbols for the main task, subtask1 and subtask2.

- SYMNOUT DD

  Listing of symbols supplied via the SYMNAMES DD.

- JNF1JMSG DD

  Messages from subtask1. This message data set will be dynamically allocated with SYSOUT=*, RECFM=FBA, LRECL=121 and BLKSIZE=121 if a JNF1JMSG DD statement is not supplied. If a JNF1JMSG DD statement is supplied, this message data set will be given the same attributes as a //SYSOUT message data set. An alternate ddname of idF1JMSG can be supplied with TASKID=id on the JOINKEYS statement for F1.

- JNF2JMSG DD

  Messages from subtask2. This message data set will be dynamically allocated with SYSOUT=*, RECFM=FBA, LRECL=121 and BLKSIZE=121 if a JNF2JMSG DD statement is not supplied. If a JNF2JMSG DD statement is supplied, this message data set will be given the same attributes as a //SYSOUT message data set. An alternate ddname of idF2JMSG can be supplied with TASKID=id on the JOINKEYS statement for F2.

- JNF1CNTL DD

  Control statements for subtask1 including INCLUDE or OMIT, OPTION, MODS, RECORD, ALTSEQ, INREC and SUM. If E35 is specified, it will not be used. The following control statements must not be specified: JOINKEYS, JOIN, REFORMAT, MERGE, OUTFIL, OUTREC or SORT.

  An alternate ddname of idF1CNTL can be supplied with TASKID=id on the JOINKEYS statement for F1.

- JNF2CNTL DD

  Control statements for subtask2 including INCLUDE or OMIT, OPTION, MODS, RECORD, ALTSEQ, INREC and SUM. If E35 is specified, it will not be used. The following control statements must not be specified: JOINKEYS, JOIN, REFORMAT, MERGE, OUTFIL, OUTREC or SORT.

  An alternate ddname of idF2CNTL can be supplied with TASKID=id on the JOINKEYS statement for F2.

- SORTWKdd DD

  Work data sets for the main task. Not recommended since dynamic allocation of work data sets is preferred. Alternate ddnames of ccccWKdd can be supplied with SORTDD=cccc in //DFSPARM.

- JNF1WKdd DD

  Work data sets for subtask1. Not recommended since dynamic allocation of work data sets will be enabled. Alternate ddnames of idF1WKdd can be supplied with TASKID=id on the JOINKEYS statement for F1.

- JNF2WKdd DD

  Work data sets for subtask2. Not recommended since dynamic allocation of work data sets will be enabled. Alternate ddnames of idF2WKdd can be supplied with TASKID=id on the JOINKEYS statement for F2.

Syntax for JOINKEYS Statements

The syntax for the JOINKEYS statements is as follows:

```
JOINKEYS {FILE=F1|FILE=F2|F1=ddname|F2=ddname}
 ,FIELDS=(p,m,s<,p,m,s>...)
 <,SORTED<,NOSEQCK>><,TYPE={F|V}><,STOPAFT=n><,TASKID=id>
 <,INCLUDE|OMIT=(logical_expression|ALL|NONE)>
```

FILES=F1 can be used as an alias for FILE=F1. FILES=F2 can be used as an alias for FILE=F2.

DFSORT symbols can be used for p,m in FIELDS, and in logical_expression in INCLUDE and OMIT wherever symbols can be used in logical_expression in COND.

Detailed Description for JOINKEYS Statement

Two JOINKEYS statements are required for a JOINKEYS application; one for the F1 file and the other for the F2 file:

- FILE=F1 or F1=ddname must be used to indicate that the JOINKEYS statement applies to the F1 input file. FILE=F1 associates the F1 file with a ddname of SORTJNF1. You can use a different ddname for the F1 file by specifying F1=ddname. For simplicity, we will use SORTJNF1 when referring to the ddname for the F1 file.
- FILE=F2 or F2=ddname must be used to indicate that the JOINKEYS statement applies to the F2 input file. FILE=F2 associates the F2 file with a ddname of SORTJNF2. You can use a different ddname for the F2 file by specifying F2=ddname. For simplicity, we will use SORTJNF2 when referring to the ddname for the F2 file.

FILE=F1 or F1=ddname

Must be used for the JOINKEYS statement associated with the F1 file. FILE=F1 (or FILES=F1) specifies a ddname of SORTJNF1 for the F1 file. F1=ddname can be used to specify any valid ddname for the F1 file. Do not use the same ddname for the F1 file and the F2 file.

When FILE=F1 or F1=ddname is specified, the other operands of the JOINKEYS statement apply to the F1 file.

FILE=F2 or F2=ddname

Must be used for the JOINKEYS statement associated with the F2 file. FILE=F2 (or FILES=F2) specifies a ddname of SORTJNF2 for the F2 file. F2=ddname can be used to specify any valid ddname for the F2 file. Do not use the same ddname for the F1 file and the F2 file.

When FILE=F2 or F2=ddname is specified, the other operands of the JOINKEYS statement apply to the F2 file.

For example, if you specify:

```
  JOINKEYS FILE=F1,FIELDS=(22,3,A),SORTED
  JOINKEYS FILE=F2,FIELDS=(15,3,A)
```

File F1 is processed using the ddname SORTJNF1 and the operands FIELDS=(22,3,A) and SORTED. File F2 is processed using the ddname SORTJNF2 and the operand FIELDS=(15,3,A).

FIELDS=(p,m,s<,p,m,s>...)

Must be specified to indicate the starting position, length and order (ascending or descending) of the keys in the input file. The keys will be treated as binary, so they must be "normalized". For example, if the keys are actually zoned decimal, they must have all C and D signs, or all F and D signs. You can use an INREC statement in JNF1CNTL and/or JNF2CNTL to normalize the keys for the F1 file and/or F2 file, respectively, if appropriate.

Each pair of keys for the F1 and F2 files must match with respect to length and order, but can start in different positions. For example, if the first key for the F1 file is 5 bytes ascending and the second key for the F1 file is 3 bytes descending, the first key for the F2 file must be 5 bytes ascending and the second key for the F2 file must be 3 bytes descending.

If a variable-length record is too short to contain a key you specify, the short key value will be compared using binary zeros for the missing bytes.

p specifies the starting position of the key. The first data byte of a fixed-length record is in position 1. The first data byte of a variable-length record is in position 5 after the 4-byte RDW. p can be 1 to 32752 but all fields must be completely contained within the first 32752 bytes of the record.

m specifies the length of the key. The total length of all keys must not exceed 4080 bytes. All fields must be completely contained within the first 32752 bytes of the record.

The length for each pair of F1 and F2 keys must match.

s specifies the order of the key. Use A for ascending or D for descending.

The order for each pair of F1 and F2 keys must match.

For example, if you specify:

```
  JOINKEYS F1=IN1,FIELDS=(22,3,A,55,9,D)
  JOINKEYS F2=IN2,FIELDS=(15,3,A,1,9,D)
```

File F1 is processed using the ddname IN1, the ascending key in positions 22-24 and the descending key in positions 55-63. File F2 is processed using the ddname IN2, the ascending key in positions 15-17 and the descending key in positions 1-9.

DFSORT symbols can be used for p,m in FIELDS.

SORTED

By default, DFSORT will sort the input file by the specified keys. If the records of the input file are already in sorted order by the specified keys, you can use the SORTED operand to tell DFSORT to copy the records rather than sort them. This can improve performance. DFSORT will terminate if the copied records are not in the order specified by the keys unless you specify the NOSEQCK operand.

For example, if you specify:

```
  JOINKEYS FILE=F1,FIELDS=(22,3,A),SORTED
  JOINKEYS FILE=F2,FIELDS=(15,3,A)
```

File F1 is copied using the ddname SORTJNF1 and the ascending key in positions 22-24. The SORTJNF1 records will be checked for the correct key order. File F2 is sorted using the ddname SORTJNF2 and the ascending key in positions 15-17.

If you use the SORTED operand, statements and options only available for a sort application, such as SUM, will be ignored for the subtask that copies the input file.

NOSEQCK

If you specify the SORTED operand and know that the records of the input file are already in sorted order by the specified keys, you can use the NOSEQCK operand to tell DFSORT not to check the order of the records. This can improve performance.

For example, if you specify:

```
  JOINKEYS FILE=F1,FIELDS=(22,3,A),SORTED,NOSEQCK
  JOINKEYS FILE=F2,FIELDS=(15,3,A),SORTED
```

File F1 is copied using the ddname SORTJNF1 and the ascending key in positions 22-24. The SORTJNF1 records will be not be checked for the correct key order. File F2 is copied using the ddname SORTJNF2 and the ascending key in positions 15-17. The SORTJNF2 records will be checked for the correct key order.

If the records are not actually in order by the specified keys and you use NOSEQCK, the output may be incorrect.

The NOSEQCK operand is ignored if the SORTED operand is not specified.

TYPE=F or TYPE=V

TYPE=V can be used to tell DFSORT to use variable-length processing for a VSAM input file. TYPE=F (the default) can be used to tell DFSORT to use fixed-length processing for a VSAM input file.

For example, if you specify:

```
  JOINKEYS F1=VSAM1,FIELDS=(22,3,A),TYPE=V
  JOINKEYS F2=VSAM2,FIELDS=(15,3,A),TYPE=F
```

VSAM file F1 is sorted as variable-length records using the ddname VSAM1 and the ascending key in positions 22-24. (Remember that for TYPE=V VSAM processing, DFSORT adds an RDW in positions 1-4 which you must account for when specifying the starting position.) VSAM file F2 is sorted as fixed-length records using the ddname VSAM2 and the ascending key in positions 15-17. (Remember that for TYPE=F VSAM processing, DFSORT does not add an RDW.)

STOPAFT=n

Can be used to specify the maximum number of records (n) you want the subtask for the input file to accept for sorting or copying (accepted means read from the input file and not deleted by INCLUDE or OMIT).

n can be up to 28 digits with up to 15 significant digits.

For example, if you specify:

```
  JOINKEYS FILE=F1,STOPAFT=5,FIELDS=(32,4,A)
  JOINKEYS FILE=F2,STOPAFT=10,FIELDS=(32,4,A)
```

The first 5 input records from SORTJNF1 and the first 10 input records from SORTJNF2 will be processed.

You can use STOPAFT=n on the OPTION statement in JNF1CNTL (for the F1 file) or JNF2CNTL (for the F2 file) instead of specifying STOPAFT=n on the JOINKEYS statement.

For example, instead of the STOPAFT operands in the JOINKEYS statements above, you could specify:

```
//SYSIN DD *
  JOINKEYS FILE=F1,FIELDS=(32,4,A)
  JOINKEYS FILE=F2,FIELDS=(32,4,A)
  ...
//JNF1CNTL DD *
  OPTION STOPAFT=5
//JNF2CNTL DD *
  OPTION STOPAFT=10
```

TASKID=id

By default, DFSORT uses the following ddnames for the subtasks:

- JNF1JMSG for the subtask1 (F1 file) message data set.
- JNF1CNTL for the subtask1 (F1 file) control data set.
- JNF1WKdd for the subtask1 (F1 file) work data sets.
- JNF2JMSG for the subtask2 (F2 file) message data set.

- JNF2CNTL for the subtask2 (F2 file) control data set.
- JNF2WKdd for the subtask2 (F2 file) work data sets.

The TASKID=id operand can be used to change the first two characters from JN to the specified id characters. The same id can be used for the F1 and F2 ddnames, or a different id can be used for each.

For example, if you specify:

```
  JOINKEYS F1=IN1,FIELDS=(1,10,A),TASKID=C1
  JOINKEYS F2=IN2,FIELDS=(22,10,A),TASKID=C1
```

C1F1JMSG, C1F1CNTL and C1F1WKdd will be used for subtask1 (F1 file). C1F2JMSG, C1F2CNTL and C1F2WKdd will be used for subtask2 (F2 file).

If you specify:

```
  JOINKEYS F1=IN1,FIELDS=(1,10,A),TASKID=I1
  JOINKEYS F2=IN2,FIELDS=(22,10,A),TASKID=I2
```

I1F1JMSG, I1F1CNTL and I1F1WKdd will be used for subtask1 (F1 file). I2F2JMSG, I2F2CNTL and I2F2WKdd will be used for subtask2 (F2 file).

The TASKID=id operand can be useful when you are doing multiple JOINKEYS applications and want to separate the messages for each application and/or specify different control statements or work data sets for different sub-tasks.

INCLUDE=(logical expression|ALL|NONE) or OMIT=(logical expression|ALL|NONE)

Can be used to specify criteria for the records you want the subtask for the input file to include or omit for sorting or copying. You can use the same logical expressions, ALL or NONE in the same way as for the INCLUDE or OMIT operand of the OUTFIL statement. See *z/OS DFSORT Application Programming Guide* for details.

For example, if you specify:

```
  JOINKEYS FILE=F1,FIELDS=(35,8,A),
     OMIT=(5,4,CH,EQ,C'ABCD')
  JOINKEYS FILE=F2,FIELDS=(37,8,A),
     INCLUDE=(1,20,SS,EQ,C'NO')
```

Only records without 'ABCD' in positions 5-8 will be processed from file F1. Only records with 'NO' somewhere in positions 1-20 will be processed from file F2.

Although the INCLUDE and OMIT operands are available on the JOINKEYS statement, it is recommended that you specify an INCLUDE or OMIT statement in JNF1CNTL or JNF2CNTL instead for ease of use.

For example, instead of the OMIT and INCLUDE operands in the JOINKEYS statements above, you could specify:

```
//SYSIN DD *
  JOINKEYS FILE=F1,FIELDS=(35,8,A)
  JOINKEYS FILE=F2,FIELDS=(37,8,A)
  ...
//JNF1CNTL DD *
  OMIT COND=(5,4,CH,EQ,C'ABCD')
//JNF2CNTL DD *
  INCLUDE COND=(1,20,SS,EQ,C'NO')
```

DFSORT symbols can be used for logical_expression in INCLUDE and OMIT wherever symbols can be used in logical_expression in COND.

Syntax for JOIN Statement

```
JOIN UNPAIRED<,F1><,F2><,ONLY>
```

Detailed Description for JOIN Statement

If you don't specify a JOIN statement for a JOINKEYS application, only paired records from F1 and F2 are kept and processed by the main task as the joined records. This is known as an inner join.

You can change which records are kept and processed by the main task as the joined records by specifying a JOIN statement. You must specify the UNPAIRED operand. The F1, F2 and ONLY operands are optional. The JOIN operands you specify indicate the joined records to be kept and processed by the main task as follows:

UNPAIRED,F1,F2 or UNPAIRED

Unpaired records from F1 and F2 as well as paired records. This is known as a full outer join.

UNPAIRED,F1

Unpaired records from F1 as well as paired records. This is known as a left outer join.

UNPAIRED,F2

Unpaired records from F2 as well as paired records. This is known as a right outer join.

UNPAIRED,F1,F2,ONLY or UNPAIRED,ONLY

Unpaired records from F1 and F2.

UNPAIRED,F1,ONLY

Unpaired records from F1.

UNPAIRED,F2,ONLY

Unpaired records from F2.

Syntax for REFORMAT Statement

```
REFORMAT FIELDS=(Fn:p,m<,p,m...><,?><,Fn:p,m<,p,m...>>...<,Fn:p>)
  <,FILL=C'c'|X'yy'>
```

DFSORT symbols can be used for p,m and p in FIELDS, and for the one-byte constant in FILL.

Detailed Description for REFORMAT Statement

A REFORMAT statement can always be used for a JOINKEYS application, and is required unless a JOIN statement with the ONLY operand is specified. The REFORMAT statement indicates the fields from the F1 file and/or the F2 file you want to include in the joined records, and the order in which you want the fields to appear. You can also include an indicator of where the key was found in the joined records ('B' for key found in F1 and F2, '1' for key found in F1 only, or '2' for key found in F2 only).

If the REFORMAT statement only defines position with length (p,m) fields, each joined record will be fixed-length (TYPE=F) with a LENGTH equal to the total length of all of the p,m fields. The maximum length for TYPE=F joined records is 32760 bytes. The F1 and F2 files can both be fixed-length, both be variable-length, or can be mixed fixed-length and variable-length.

If the REFORMAT statement defines a position without a length (p without m) field, each joined record will be variable-length (TYPE=V) with a LENGTH equal to the total length of all of the p,m fields plus the variable length of each p field (one for F1 and/or one for F2). The maximum length for TYPE=V joined records is 32767 bytes. The F1 and F2 files can both be variable-length or can be mixed fixed-length and variable-length. However:

- The first field must contain the RDW (1,n with n equal to or greater than 4) and must be from a variable-length file (F1 or F2).
- The position without length fields (one from the F1 file and/or one from the F2 file) must be the last fields specified and must be from a variable-length file (F1 and/or F2).

For joined records created from paired records, any F1 fields specified will be extracted from the F1 record and any F2 fields specified will be extracted from the F2 record.

For joined records created from unpaired F1 records, any F1 fields specified will be extracted from the F1 record and any F2 fields specified will be filled with the specified FILL character or blanks by default. However, if the F2 file is variable-length, any specified F2 RDW fields (1,n with n equal to or less than 4) will be filled with binary zeros.

For joined records created from unpaired F2 records, any F2 fields specified will be extracted from the F2 record and any F1 fields specified will be filled with the specified FILL character or blanks by default. However, if the F1 file is variable-length, any specified F1 RDW fields (1,n with n equal to or less than 4) will be filled with binary zeros.

If a JOIN statement with the ONLY operand is specified, the REFORMAT statement does not have to be specified. In this case, the layout of the joined records will depend on the specified JOIN statement operands as follows:

- JOIN UNPAIRED,F1,ONLY

  The joined records will be the original unpaired F1 records. If the F1 records are fixed-length, the joined records will be fixed-length. If the F1 records are variable-length, the joined records will be variable-length.

- JOIN UNPAIRED,F2,ONLY

  The joined records will be the original unpaired F2 records. If the F2 records are fixed-length, the joined records will be fixed-length. If the F2 records are variable-length, the joined records will be variable-length.

- JOIN UNPAIRED,F1,F2,ONLY or JOIN UNPAIRED,ONLY

  The joined records will be variable-length. If the F1 records are fixed-length, each unpaired F1 record will be variable-length with an RDW followed by the original F1 record. If the F1 records are variable-length, each unpaired F1 record will be the original F1 record. If the F2 records are fixed-length, each unpaired F2 record will be variable-length with an RDW followed by the original F2 record. If the F2 records are variable-length, each unpaired F2 record will be the original F2 record.

In all cases, the TYPE and LENGTH will be set as appropriate for the joined records.

For joined records with TYPE=F, the maximum LENGTH is 32760.

For joined records with TYPE=V, the maximum LENGTH is 32767. Note that the RECFM of the output file must be VS or VBS in order to allow output records greater than 32756.

FIELDS=(Fn:p,m<,p,m...><,?><,Fn:p,m<,p,m...>>...<,Fn:p>)

Must be specified to indicate the starting position and length of each field from the F1 file and/or the F2 file to be included in the joined records, and optionally an indicator of where the key was found. The fields and indicator will be included in the joined records in the order in which they are specified.

F1: indicates the following fields up to the next Fn: or end of the FIELDS operand are from the F1 file.

F2: indicates the following fields up to the next Fn: or end of the FIELDS operand are from the F2 file.

p,m specifies the starting position and length of a fixed field. p specifies the starting position of the field. The first data byte of a fixed-length record is in position 1. The first data byte of a variable-length record is in position 5 after the 4-byte RDW. p can be 1 to 32767. m specifies the length of the field. m can be 1 to 32760. All fields must be completely contained within the first 32767 bytes of the record.

? indicates a 1-byte indicator is to be included in each joined record. The indicator will be set to one of the following values in each paired or unpaired joined record, as appropriate:

- 'B' - the key was found in F1 and F2.
- '1' - the key was found in F1, but not in F2.
- '2' - the key was found in F2, but not in F1.

Only one ? can be specified in the FIELDS operand. If ? is not the last item, it must be followed by F1: or F2:.

For TYPE=F joined records, the indicator can appear anywhere in the record. For example, each of the following is valid:

```
* Put indicator in position 1 of each joined record.
  REFORMAT FIELDS=(?,F1:1,20,F2:5,8)

* Put indicator in position 21 of each joined record.
  REFORMAT FIELDS=(F1:1,20,?,F1:31,9)

* Put indicator in position 25 of each joined record.
  REFORMAT FIELDS=(F2:11,20,6,4,?)
```

For TYPE=V joined records, the indicator must appear in the fixed part of the record, that is, after the RDW and before the position without length fields. For example, the following is valid:

```
* Put indicator in position 5 of each joined record.
  REFORMAT FIELDS=(F1:1,4,?,F1:11)
```

p (without m) gives the starting position of a variable field. Only one p without m field can be specified for F1 and only one p without m field can be specified for F2.

If either or both p without m fields are specified, they must be the last fields in the FIELDS operand. In this case, the first field in the FIELDS operand must be from a variable-length file (F1 or F2) and must include the RDW (1,n where n is equal to or greater than 4).

Example with just p,m fields:

```
  REFORMAT FIELDS=(F1:27,5,1,8,F2:19,20,F1:1201,15)
```

Example with one p without m field:

```
  REFORMAT FIELDS=(F1:1,4,F2:6,25,92,2,F1:8,9,32)
```

Example with two p without m fields:

```
  REFORMAT FIELDS=(F2:1,9,21,3,F1:101,7,28,9,122,F2:26)
```

FILL=C'c' or FILL=X'yy'

The FILL operand can be used to override DFSORT's default fill byte of a blank (X'40'). The fill byte is used in the following situations:

- A p,m (fixed) field is specified for a file (F1 or F2) with variable-length records, and the field extends beyond the end of a record. Each missing byte is replaced with the fill byte. For example, if a variable-length F1 record is 20 bytes long and the REFORMAT statement has:

  ```
  REFORMAT FIELDS=(F1:1,30,41),FILL=C'*'
  ```

  bytes 21-30 of the joined record are filled with asterisks.

- The REFORMAT statement has a p,m (fixed) field from the F1 file and a joined record is being created from an unpaired F2 record. For example if the following are specified:

  ```
  JOIN UNPAIRED,F2
  REFORMAT FIELDS=(F1:21,5,F2:1,10),FILL=X'00'
  ```

  and an unpaired F2 record is found, the joined record will have 5 binary zero bytes followed by bytes 1-10 from the F2 record.

- The REFORMAT statement has a p,m (fixed) field from the F2 file and a joined record is being created from an unpaired F1 record. For example if the following are specified:

  ```
  JOIN UNPAIRED,F1,F2
  REFORMAT FIELDS=(F1:21,5,F2:1,10),FILL=X'00'
  ```

  and an unpaired F1 record is found, the joined record will have bytes 21-25 of the F1 record followed by 10 binary zero bytes. (Since UNPAIRED,F1,F2 is specified, if an unpaired F2 record is found, the joined record will have 5 binary zero bytes followed by bytes 1-10 of the F2 record.)

C'c' specifies a character fill byte. c must be one EBCDIC character. If you want to use an apostrophe as the fill byte, you must specify it as C''''.

X'yy' specifies a hexadecimal fill byte. yy must be one pair of hexadecimal digits (00-FF).

JOINKEYS Application Notes

In the notes below, "subtaskn" refers to subtask1 for the F1 file or subtask2 for the F2 file.

Notes:

1. Since a JOINKEYS application uses three tasks, it can require more storage than a regular DFSORT application. You may need to use REGION=0M for some JOINKEYS applications.
2. DFSORT's normal syntax and continuation rules are used for the JOINKEYS, JOIN and REFORMAT statements. See *z/OS DFSORT Application Programming Guide* for details.
3. Control statements from DFSPARM, SYSIN (direct invocation) or SORTCNTL (program invocation) will be used for the main task, but not for the subtasks. Control statements from JNF1CNTL will be used for subtask1. Control statements from JNF2CNTL will be used for subtask2.
4. For the main task, the normal options in effect for a DFSORT run will be used. The run-time options will be those from DFSPARM, SYSIN, SORTCNTL, EXEC PARM or the caller's parameter list, as appropriate. The installation options will be those for direct invocation or program invocation of DFSORT, as appropriate.

5. For subtaskn, the options in effect for the subtask will be used with the exceptions noted below. The run-time options will be those from JNFnCNTL. The installation options will be those for a calling program (INV, TSOINV, TD1-TD4).
6. TYPE=F processing is used for a VSAM F1 or F2 file by default. TYPE=V can be specified in JNFnCNTL, or on the JOINKEYS statement for Fn, to override the default of TYPE=F.
7. For subtaskn, if an INCLUDE or OMIT statement or a STOPAFT or TYPE option is specified in JNFnCNTL, it will override the corresponding option in a JOINKEYS statement for Fn. For ease of use, it is recommended that you use an INCLUDE or OMIT statement, and a STOPAFT or TYPE option, in JNFnCNTL rather than an INCLUDE, OMIT, STOPAFT or TYPE operand in a JOINKEYS statement.
8. For the main task, override of statements and options is done in the normal way, that is, DFSPARM overrides SYSIN for direct invocation of DFSORT, and DFSPARM overrides SORTCNTL overrides the caller's parameter list for program invocation of DFSORT. See *z/OS DFSORT Application Programming Guide* for details.
9. The following options will be used for subtaskn and cannot be overridden: LIST, MSGPRT=ALL, NOABEND, ESTAE, NOCHECK, EQUALS and RESINV=0. DYNALLOC will be used automatically for subtaskn, but can be overridden by a DYNALLOC option in JNFnCNTL.
10. If an INCLUDE or OMIT statement is specified in JNFnCNTL, DFSORT will use the following options in effect for subtaskn processing: ALTSEQ, CHALT or NOCHALT, SZERO or NOSZERO and VLSCMP or NOVLSCMP. These options can be specified in JNFnCNTL. The subtaskn LOCALE installation option in effect will also be used if appropriate.
11. If an INCLUDE or OMIT operand is specified on the JOINKEYS statement for Fn, DFSORT will use these main task options in effect for subtaskn processing: ALTSEQ, CHALT or NOCHALT, SZERO or NOSZERO and VLSCMP or NOVLSCMP. If these options are specified in JNFnCNTL, they will be ignored for subtaskn processing. LOCALE will not be used.
12. For subtaskn, the following statements are not allowed in JNFnCNTL: JOINKEYS, JOIN, MERGE, OUTFIL, OUTREC, REFORMAT and SORT.
13. For the main task, a MERGE statement other than MERGE FIELDS=COPY is not allowed in any source.
14. If appropriate, options such as FILSZ, DYNALLOC, AVGRLEN, and so on can be specified in JNFnCNTL for subtaskn, or in DFSPARM for the main task.
15. If tape data sets are used for the F1 and F2 files, they must be on different drives so DFSORT can read them in parallel.

Example 1 - Paired F1/F2 records without duplicates

```
//JKE1  EXEC  PGM=SORT
//SYSOUT    DD  SYSOUT=*
//SORTJNF1 DD *
Roses          03  Red
Daisies        06  Orange
Roses          04  Pink
Daisies        02  Yellow
Roses          06  Yellow
Daisies        12  Lilac
Roses          09  Blue
/*
//SORTJNF2 DD *
Red      Lilies          InStock
Red      Roses           InStock
Orange   Daisies         SoldOut
Pink     Roses           SoldOut
Yellow   Daisies         InStock
Yellow   Roses           Ordered
Lilac    Daisies         SoldOut
White    Daisies         InStock
/*
//SORTOUT DD SYSOUT=*
//SYSIN    DD    *
* Control statements for JOINKEYS application
  JOINKEYS FILE=F1,FIELDS=(1,15,A,20,8,A)
  JOINKEYS FILE=F2,FIELDS=(10,15,A,1,8,A)
  REFORMAT FIELDS=(F1:20,8,1,15,F2:26,10,F1:16,2)
* Control statements for main task (joined records)
  OPTION COPY
  OUTFIL REMOVECC,
    HEADER2=(1:'Color',11:'Flower',26:'Status',36:'Per Pot',/,
             1:7'-',11:14'-',26:7'-',36:7'-'),
    BUILD=(1:1,8,11:9,15,26:24,10,
      36:34,2,ZD,M10,LENGTH=7)
/*
```

This example illustrates how you can join paired records from two files using multiple keys. In this case, neither file has duplicates. The paired records are the records in F1 and F2 with matching keys (for example, key1=Roses and key2=Red).

Input file1 (F1) has RECFM=FB and LRECL=80. It contains the records shown for SORTJNF1 in the JCL above. Input file2 (F2) has RECFM=FB and LRECL=80. It contains the records shown for SORTJNF2 in the JCL above.

The output file will have RECFM=FB and LRECL=42. It will contain the paired records from the two files reformatted as follows:

```
Color     Flower         Status    Per Pot
-------   -------------- -------   -------
Lilac     Daisies        SoldOut        12
Orange    Daisies        SoldOut         6
Yellow    Daisies        InStock         2
Pink      Roses          SoldOut         4
Red       Roses          InStock         3
Yellow    Roses          Ordered         6
```

The first JOINKEYS statement defines the ddname and keys for the F1 file. FILE=F1 tells DFSORT that the ddname for the F1 file is SORTJNF1. FIELDS=(1,15,A,20,8,A) tells DFSORT that the first binary key is in positions 1-15 ascending and the second binary key is in positions 20-27 ascending. Since SORTED is not specified, DFSORT will sort the SORTJNF1 records by the specified binary keys.

The second JOINKEYS statement defines the ddname and keys for the F2 file. FILE=F2 tells DFSORT that the ddname for the F2 file is SORTJNF2. FIELDS=(10,15,A,1,8,A) tells DFSORT that the first binary key is in positions 10-24 ascending and the second binary key is in positions 1-8 ascending. Since SORTED is not specified, DFSORT will sort the SORTJNF2 records by the specified binary keys.

Note that corresponding keys for the two files match in length and order.

The REFORMAT statement defines the fields to be extracted for the joined records in the order in which they are to appear. FIELDS=(F1:20,8,1,15,F2:26,10,F1:16,2) tells DFSORT to create the joined records as follows:

```
Joined Record Positions     Extracted from
-----------------------     ------------------
1-8                         F1 positions 20-27
9-23                        F1 positions 1-15
24-33                       F2 positions 26-35
34-35                       F1 positions 16-17
```

Since there is no JOIN statement, only paired records are joined by default.

The OPTION COPY statement tells DFSORT to copy the joined records. The OUTFIL statement tells DFSORT to reformat the joined records, display a header at the top of each page and remove the carriage control characters. Note that the BUILD operand of the OUTFIL statement must reference the positions of fields in the joined records.

Conceptually, JOINKEYS application processing proceeds as follows:

- Subtask1 sorts the SORTJNF1 (F1 file) records as directed by its JOINKEYS statement. As a result, it passes the following records to the main task:

```
Daisies        12  Lilac
Daisies        06  Orange
Daisies        02  Yellow
Roses          09  Blue
Roses          04  Pink
Roses          03  Red
Roses          06  Yellow
```

- Subtask2 sorts the SORTJNF2 (F2 file) records as directed by its JOINKEYS statement. As a result, it passes the following records to the main task:

```
Lilac    Daisies        SoldOut
Orange   Daisies        SoldOut
White    Daisies        InStock
Yellow   Daisies        InStock
Red      Lilies         InStock
Pink     Roses          SoldOut
Red      Roses          InStock
Yellow   Roses          Ordered
```

- The main task joins the records passed from subtask1 and subtask2 as directed by the specified JOINKEYS and REFORMAT statements, resulting in the following joined records:

```
Lilac   Daisies        SoldOut   12
Orange  Daisies        SoldOut   06
Yellow  Daisies        InStock   02
Pink    Roses          SoldOut   04
Red     Roses          InStock   03
Yellow  Roses          Ordered   06
```

- Finally, the main task copies and reformats the joined records according to the OUTFIL statement, and writes the resulting records to SORTOUT. Thus, SORTOUT contains these records:

```
Color     Flower         Status    Per Pot
-------   -------------- -------   -------
Lilac     Daisies        SoldOut        12
Orange    Daisies        SoldOut         6
Yellow    Daisies        InStock         2
Pink      Roses          SoldOut         4
Red       Roses          InStock         3
Yellow    Roses          Ordered         6
```

Example 2 - Paired F1/F2 records with duplicates (cartesian)

```
//JKE2  EXEC  PGM=SORT
//SYSOUT    DD  SYSOUT=*
//VBIN DD DSN=MY.VBFILE,DISP=SHR
//FBIN DD DSN=MY.FBFILE,DISP=SHR
//SORTOUT DD DSN=MY.FB.OUTPUT,DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,(5,5)),UNIT=SYSDA
//SYSIN    DD    *
* Control statements for JOINKEYS application
  JOINKEYS F1=VBIN,FIELDS=(18,16,A),SORTED
  JOINKEYS F2=FBIN,FIELDS=(1,16,A)
  REFORMAT FIELDS=(F2:22,12,F1:5,12,F2:1,16)
* Control statements for main task (joined records)
  OPTION EQUALS
  SORT FIELDS=(13,12,CH,A)
/*
```

This example illustrates how you can join paired records from two files, both of which have duplicate records. The result will be a cartesian join. The paired records are the records in F1 and F2 with matching keys (for example, key=Cats).

Input file1 has RECFM=VB and LRECL=50. It contains the following records:

```
Len|Data
 40|Eliot        Cats            Musical
 40|Lloyd-Webber Cats            Musical
 48|Hart         Pal Joey        Musical, Comedy
 48|Rodgers      Pal Joey        Musical, Comedy
 47|Hammerstein  South Pacific   Musical, Drama
 47|Rodgers      South Pacific   Musical, Drama
```

Input file2 has RECFM=FB and LRECL=36. It contains the following records:

```
Pal Joey            Start: 1940  22
South Pacific       Start: 1949  13
Cats                Start: 1982  50
South Pacific       End:   1954
Cats                End:   2000
Pal Joey            End:   1941
```

The output file will have RECFM=FB and LRECL=40. It will contain the paired cartesian product of the two files sorted as follows:

```
Start: 1982 Eliot       Cats
End:   2000 Eliot       Cats
Start: 1949 Hammerstein South Pacific
End:   1954 Hammerstein South Pacific
Start: 1940 Hart        Pal Joey
End:   1941 Hart        Pal Joey
Start: 1982 Lloyd-WebberCats
End:   2000 Lloyd-WebberCats
Start: 1940 Rodgers     Pal Joey
End:   1941 Rodgers     Pal Joey
Start: 1949 Rodgers     South Pacific
End:   1954 Rodgers     South Pacific
```

The first JOINKEYS statement defines the ddname and key for the F1 file. F1=VBIN tells DFSORT that the ddname for the F1 file is VBIN. FIELDS=(18,16,A) tells DFSORT that the key is in positions 18-33 ascending. Note that since VBIN is a VB file, the starting position of its key must take the RDW in positions 1-4 into account. Since SORTED is specified, indicating that the records are already in order by the specified binary key, DFSORT will copy the VBIN records.

The second JOINKEYS statement defines the ddname and binary key for the F2 file. F2=FBIN tells DFSORT that the ddname for the F2 file is FBIN. FIELDS=(1,16,A) tells DFSORT that the binary key is in positions 1-16 ascending. Since SORTED is not specified, DFSORT will sort the FBIN records by the specified binary key.

The REFORMAT statement defines the fields to be extracted for the joined records in the order in which they are to appear. FIELDS=(F2:22,12,F1:5,12,F2:1,16) tells DFSORT to create the joined records as follows:

```
Joined Record Positions     Extracted from
-----------------------     ------------------
1-12                        F2 positions 22-33
13-24                       F1 positions 5-16
25-40                       F2 positions 1-16
```

Note that since VBIN (F1) is a VB file, the starting position of its REFORMAT field must take the RDW in positions 1-4 into account.

Since there is no JOIN statement, only paired records are joined by default. Since there are duplicates in each input file, a cartesian join is performed.

The SORT FIELDS=(13,12,CH,A) statement tells DFSORT to sort the joined records by a different key than the one used for the join of F1 and F2 records. Note that the FIELDS operand of the SORT statement must reference the positions of fields in the joined records.

Conceptually, JOINKEYS application processing proceeds as follows:

- Subtask1 copies the VBIN (F1 file) records as directed by its JOINKEYS statement. As a result, it passes the following records to the main task:

```
Len|Data
 40|Eliot        Cats            Musical
 40|Lloyd-Webber Cats            Musical
 48|Hart         Pal Joey        Musical, Comedy
 48|Rodgers      Pal Joey        Musical, Comedy
 47|Hammerstein  South Pacific   Musical, Drama
 47|Rodgers      South Pacific   Musical, Drama
```

- Subtask2 sorts the FBIN (F2 file) records as directed by its JOINKEYS statement. As a result, it passes the following records to the main task:

```
Cats                 Start: 1982  50
Cats                 End:   2000
Pal Joey             Start: 1940  22
Pal Joey             End:   1941
South Pacific        Start: 1949  13
South Pacific        End:   1954
```

- The main task joins the records passed from subtask1 and subtask2 as directed by the specified JOINKEYS and REFORMAT statements, resulting in the following joined records:

```
Start: 1982 Eliot       Cats
End:   2000 Eliot       Cats
Start: 1982 Lloyd-WebberCats
End:   2000 Lloyd-WebberCats
Start: 1940 Hart        Pal Joey
End:   1941 Hart        Pal Joey
Start: 1940 Rodgers     Pal Joey
End:   1941 Rodgers     Pal Joey
Start: 1949 Hammerstein South Pacific
End:   1954 Hammerstein South Pacific
Start: 1949 Rodgers     South Pacific
End:   1954 Rodgers     South Pacific
```

- Finally, the main task sorts the joined records according to the SORT statement, and writes the resulting records to SORTOUT. Thus, SORTOUT contains these records:

```
Start: 1982 Eliot       Cats
End:   2000 Eliot       Cats
Start: 1949 Hammerstein South Pacific
End:   1954 Hammerstein South Pacific
Start: 1940 Hart        Pal Joey
End:   1941 Hart        Pal Joey
Start: 1982 Lloyd-WebberCats
End:   2000 Lloyd-WebberCats
Start: 1940 Rodgers     Pal Joey
End:   1941 Rodgers     Pal Joey
Start: 1949 Rodgers     South Pacific
End:   1954 Rodgers     South Pacific
```

Example 3 - Paired F1 records

```
//JKE3  EXEC  PGM=SORT
//SYSOUT    DD  SYSOUT=*
//MASTER DD DSN=MASTER.FILE,DISP=SHR
//PULL DD DSN=PULL.FILE,DISP=SHR
//SORTOUT DD SYSOUT=*
//JNF1CNTL DD *
* Control statement for subtask1 (F1)
  INREC PARSE=(%01=(ENDBEFR=C',',FIXLEN=15),
              %=(ENDBEFR=C','),
              %02=(ENDBEFR=C',',FIXLEN=10)),
  OVERLAY=(36:%01,52:%02)
/*
//DFSPARM   DD    *
* Control statements for JOINKEYS application
  JOINKEYS F1=MASTER,FIELDS=(36,15,A,52,10,A)
  JOINKEYS F2=PULL,FIELDS=(12,15,A,1,10,A)
  REFORMAT FIELDS=(F1:1,35)
* Control statement for main task
  OPTION COPY
/*
```

This example illustrates how you can select only paired records from one of two files. In this case, we will select the F1 records that have a match in F2 on the specified keys (for example, key1=Development and key2=Master). The F1 records are in comma delimited form so we will parse them to get the binary keys we need.

Input file1 (F1) has RECFM=FB and LRECL=35. It contains the following records:

```
Marketing,William,Master
Development,John,Bachelor
Manufacturing,Louis,Master
Development,Carol,Master
Research,Angela,Master
Research,Anne,Doctorate
Development,Sara,Doctorate
Marketing,Joseph,Master
Manufacturing,Donna,Bachelor
Development,Susan,Master
Manufacturing,Nick,Master
Research,Howard,Doctorate
```

Input file2 (F2) has RECFM=FB and LRECL=30. It contains the following records:

```
Master     Development
Master     Manufacturing
Bachelor   Development
Doctorate  Research
```

The output file will have RECFM=FB and LRECL=35. It will contain the paired F1 records, that is, the records from F1 that have a match in F2 for the specified keys (the first and third fields):

```
Development,John,Bachelor
Development,Carol,Master
Development,Susan,Master
Manufacturing,Louis,Master
Manufacturing,Nick,Master
Research,Anne,Doctorate
Research,Howard,Doctorate
```

The first JOINKEYS statement defines the ddname and keys for the F1 file. F1=MASTER tells DFSORT that the ddname for the F1 file is MASTER.

The control statements in JNF1CNTL will be used to process the F1 file before it is sorted. The input records are in comma delimited form, so to use the first and third fields as binary keys, while preserving the original data, we use an INREC statement to parse out the fields we want and add them to the end of the record. The parsed first key will be in positions 36-50 and the parsed second key will be in positions 52-61. For example, the first reformatted record will look like this:

```
Marketing,William,Master           Marketing       Master
```

FIELDS=(36,15,A,52,10,A) in the JOINKEYS statement (referring to the reformatted INREC positions) tells DFSORT that the first key is in positions 36-50 ascending and the second key is in positions 52-61 ascending.

The second JOINKEYS statement defines the ddname and keys for the F2 file. F2=PULL tells DFSORT that the ddname for the F2 file is PULL.

FIELDS=(12,15,A,1,10,A) in the JOINKEYS statement tells DFSORT that the first key is in positions 12-26 ascending and the second key is in positions 1-10 ascending.

The REFORMAT statement defines the fields to be extracted for the joined records in the order in which they are to appear. FIELDS=(F1:1,35) tells DFSORT to create the joined records from the original F1 input records (positions 1-35 of the F1 records). Since we only needed the added parsed fields for sorting, we don't need to include them in the joined records. Of course, if we wanted the main task to "see" the parsed fields in the joined records, we could include the parsed fields in the REFORMAT FIELDS operand.

Since there is no JOIN statement, only paired records are joined by default.

The OPTION COPY statement tells DFSORT to copy the joined records.

Conceptually, JOINKEYS application processing proceeds as follows:

- Subtask1 performs INREC processing for the MASTER (F1 file) records as directed by the control statement in JNF1CNTL and sorts the resulting records as directed by its JOINKEYS statement. As a result, it passes the following records to the main task:

```
Development,John,Bachelor          Development     Bachelor
Development,Sara,Doctorate         Development     Doctorate
Development,Carol,Master           Development     Master
Development,Susan,Master           Development     Master
Manufacturing,Donna,Bachelor       Manufacturing   Bachelor
Manufacturing,Louis,Master         Manufacturing   Master
Manufacturing,Nick,Master          Manufacturing   Master
Marketing,William,Master           Marketing       Master
Marketing,Joseph,Master            Marketing       Master
Research,Howard,Doctorate          Research        Doctorate
Research,Anne,Doctorate            Research        Doctorate
Research,Angela,Master             Research        Master
```

- Subtask2 sorts the PULL (F2 file) records as directed by its JOINKEYS statement. As a result, it passes the following records to the main task:

```
Bachelor   Development
Master     Development
Master     Manufacturing
Doctorate  Research
```

- The main task joins the records passed from subtask1 and subtask2 as directed by the specified JOINKEYS and REFORMAT statements, resulting in the following joined records:

  ```
  Development,John,Bachelor
  Development,Carol,Master
  Development,Susan,Master
  Manufacturing,Louis,Master
  Manufacturing,Nick,Master
  Research,Anne,Doctorate
  Research,Howard,Doctorate
  ```

- Finally, the main task copies the joined records to SORTOUT. Thus, SORTOUT contains these records:

  ```
  Development,John,Bachelor
  Development,Carol,Master
  Development,Susan,Master
  Manufacturing,Louis,Master
  Manufacturing,Nick,Master
  Research,Anne,Doctorate
  Research,Howard,Doctorate
  ```

Example 4 - Unpaired F2 records

```
//JKE4  EXEC  PGM=SORT
//SYSOUT    DD  SYSOUT=*
//IN1 DD DSN=FILE1.IN,DISP=SHR
//IN2 DD DSN=FILE2.IN,DISP=SHR
//SORTOUT DD DSN=FILE3.OUT,DISP=OLD
//JNF1CNTL DD *
* Control statements for subtask1 (F1)
  OMIT COND=(10,5,UFF,EQ,99999)
  INREC BUILD=(1,8,9:10,5,UFF,TO=ZD,LENGTH=5)
/*
//JNF2CNTL DD *
* Control statements for subtask2 (F2)
  OMIT COND=(14,5,UFF,EQ,99999)
  INREC BUILD=(1,4,5:14,5,UFF,TO=ZD,LENGTH=5,10:5)
/*
//SYSIN    DD    *
* Control statements for JOINKEYS application
  JOINKEYS F1=IN1,FIELDS=(1,8,A,9,5,D)
  JOINKEYS F2=IN2,FIELDS=(10,8,A,5,5,D)
  JOIN UNPAIRED,F2,ONLY
  REFORMAT FIELDS=(F2:1,4,10)
* Control statement for main task
  OPTION COPY
/*
```

This example illustrates how you can select only unpaired records from one of two files. In this case, we will select the F2 records that do not have a match in F1 on the specified keys (for example, key1=Molly and key2=2100). We will also omit certain records from each input file and handle unnormalized keys.

Input file1 has RECFM=FB and LRECL=15. It contains the following records:

```
Molly    145
Molly    99999
Molly    2143
Jasmine  1292
Jasmine  5
Jasmine  28
Jasmine  99999
```

Input file2 has RECFM=VB and LRECL=35. It contains the following records:

```
Len|Data
 30|Molly    145      Thursday
 31|Molly    2100     Wednesday
 28|Molly    18       Sunday
 28|Jasmine  99999    Monday
 28|Jasmine  5        Sunday
 30|Jasmine  28       Saturday
 29|Jasmine  103      Tuesday
 31|Jasmine  99999    Wednesday
```

The output file will have RECFM=VB and LRECL=35. F1 records with 99999 in positions 10-14 and F2 records with 99999 in positions 14-18 will be removed before join processing. The output file will contain unpaired F2 records (that is, records from F2 that do not have a match in F1 for the specified keys) as follows:

```
Len|Data
 29|Jasmine  103      Tuesday
 31|Molly    2100     Wednesday
 28|Molly    18       Sunday
```

The first JOINKEYS statement defines the ddname and keys for the F1 file. F1=IN1 tells DFSORT that the ddname for the F1 file is IN1.

The control statements in JNF1CNTL will be used to process the F1 file before it is sorted. The OMIT statement removes records that have 99999 in positions 10-14. We want to use the numeric field as our key. However, the numeric field is unnormalized since it is left aligned instead of right aligned, so sorting it as a binary key will not work. To handle this, we use the INREC statement to reformat the numeric field as ZD values in positions 9-13 (positive ZD values with the same sign can be sorted as binary). For example, the first reformatted FB record will look like this:

```
Molly   00145
```

Since we don't need the F1 records for output, we don't need to keep the original left aligned numeric value.

FIELDS=(1,8,A,9,5,D) in the JOINKEYS statement (referring to the reformatted INREC positions) tells DFSORT that the first key is in positions 1-8 ascending and the second key is in positions 9-13 descending.

The second JOINKEYS statement defines the ddname and keys for the F2 file. F2=IN2 tells DFSORT that the ddname for the F2 file is IN2.

The control statements in JNF2CNTL will be used to process the F2 file before it is sorted. The OMIT statement removes records that have 99999 in positions 14-18. Again, we need a ZD version of the left aligned numeric value to use for the binary key. But in this case, since we want the original F2 records for output, we need to keep the original numeric value as well. Using the INREC statement, we add the ZD value at positions 5-9 between the RDW and the first data field. That shifts the original data to start at position 10. For example, the first reformatted VB record will look like this:

```
Len|Data
 35|00145Molly    145      Thursday
```

In this case, since the input is a VB file, we specify the RDW (1,4), then the converted field, and then the rest of the record (5 without a length) in the INREC statement.

FIELDS=(10,8,A,5,5,D) in the JOINKEYS statement (referring to the reformatted INREC positions) tells DFSORT that the first key is in positions 10-17 ascending and the second key is in positions 5-9 descending.

Note that since IN2 is a VB file, all of its starting positions must take the RDW in positions 1-4 into account.

The JOIN statement tells DFSORT that the joined records should be the F2 records that do not have a match in F1 on the specified keys.

The REFORMAT statement defines the fields to be extracted for the joined records in the order in which they are to appear. We need the RDW (1,4) and the original data which starts in position 10 of the reformatted F2 records. So we use FIELDS=(F2:1,4,10). Since the last field (10) is a position without a length, it tells DFSORT to create VB records. The joined records are created as follows from the reformatted F2 records:

```
Joined Record Positions     Extracted from
-----------------------     ----------------------------------
1-4                         RDW (not extracted)
5 to end                    Reformatted F2 position 10 to end
```

Conceptually, JOINKEYS application processing proceeds as follows:

- Subtask1 performs OMIT and INREC processing for the IN1 (F1 file) records as directed by the control statements in JNF1CNTL and sorts the resulting records as directed by its JOINKEYS statement. As a result, it passes the following records to the main task:

```
Jasmine 01292
Jasmine 00028
Jasmine 00005
Molly   02143
Molly   00145
```

- Subtask2 performs OMIT and INREC processing for the IN2 (F2 file) records as directed by the control statements in JNF2CNTL and sorts the resulting records as directed by its JOINKEYS statement. As a result, it passes the following records to the main task:

```
Len|Data
 34|00103Jasmine  103      Tuesday
 35|00028Jasmine  28       Saturday
 33|00005Jasmine  5        Sunday
 36|02100Molly    2100     Wednesday
 35|00145Molly    145      Thursday
 33|00018Molly    18       Sunday
```

- The main task joins the records passed from subtask1 and subtask2 as directed by the specified JOINKEYS, JOIN and REFORMAT statements, resulting in the following joined records (unmatched F2 records):

```
Len|Data
 29|Jasmine  103      Tuesday
 31|Molly    2100     Wednesday
 28|Molly    18       Sunday
```

- Finally, the main task copies the joined records, and writes them to SORTOUT. Thus, SORTOUT contains these records:

```
Len|Data
 29|Jasmine  103      Tuesday
 31|Molly    2100     Wednesday
 28|Molly    18       Sunday
```

Example 5 - Paired and unpaired F1/F2 records (indicator method)

```
//JKE5  EXEC  PGM=SORT
//SYSOUT    DD  SYSOUT=*
//SORTJNF1 DD DSN=FIRST.FILE,DISP=SHR
//SORTJNF2 DD DSN=SECOND.FILE,DISP=SHR
//F1ONLY DD SYSOUT=*
//F2ONLY DD SYSOUT=*
//BOTH DD SYSOUT=*
//SYSIN    DD    *
* Control statements for JOINKEYS application
  JOINKEYS FILE=F1,FIELDS=(1,10,A),SORTED,NOSEQCK
  JOINKEYS FILE=F2,FIELDS=(7,10,A),SORTED,NOSEQCK
  JOIN UNPAIRED,F1,F2
  REFORMAT FIELDS=(F1:1,14,F2:1,20,?)
* Control statements for main task (joined records)
  OPTION COPY
  OUTFIL FNAMES=F1ONLY,INCLUDE=(35,1,CH,EQ,C'1'),
    BUILD=(1,14)
  OUTFIL FNAMES=F2ONLY,INCLUDE=(35,1,CH,EQ,C'2'),
    BUILD=(15,20)
  OUTFIL FNAMES=BOTH,INCLUDE=(35,1,CH,EQ,C'B'),
    BUILD=(1,14,/,15,20)
/*
```

This example illustrates how you can create three output files; with paired F1/F2 records, unpaired F1 records and unpaired F2 records. In this case:

- F1 records which do not have a match in F2 on the specified keys (for example, key=David) will be written to the F1ONLY output file.
- F2 records which do not have a match in F1 on the specified keys (for example, key=Karen) will be written to the F2ONLY output file.
- F1 and F2 records which have a match in F1 and F2 on the specified keys (for example, key=Carrie) will be written to the BOTH output file.

Input file1 has RECFM=FB and LRECL=14. It contains the following records:

```
Carrie    F101
David     F102
Frank     F103
Holly     F104
Vicky     F105
```

Input file2 has RECFM=FB and LRECL=20. It contains the following records:

```
No    Carrie    F201
Yes   Holly     F202
Yes   Karen     F203
No    Sri Hari  F204
Yes   Vicky     F205
```

The F1ONLY file will have RECFM=FB and LRECL=14. It will contain the unpaired F1 records as follows:

```
David     F102
Frank     F103
```

The F2ONLY file will have RECFM=FB and LRECL=20. It will contain the unpaired F2 records as follows:

```
Yes   Karen     F203
No    Sri Hari  F204
```

The BOTH file will have RECFM=FB and LRECL=20. It will contain the paired F1 and F2 records as follows:

```
Carrie    F101
No    Carrie    F201
Holly     F104
Yes   Holly     F202
Vicky     F105
Yes   Vicky     F205
```

The first JOINKEYS statement defines the ddname and key for the F1 file. FILE=F1 tells DFSORT that the ddname for the F1 file is SORTJNF1. FIELDS=(1,10,A) tells DFSORT that the key is in positions 1-10 ascending. Since SORTED is specified, indicating that the records are already in order by the specified binary key, DFSORT will copy the SORTJNF1 records. Since NOSEQCK is specified, DFSORT will not check that the records are in order by the key. (Only use NOSEQCK if you know for sure that the records are in order by the key.)

The second JOINKEYS statement defines the ddname and key for the F2 file. FILE=F2 tells DFSORT that the ddname for the F2 file is SORTJNF2. FIELDS=(7,10,A) tells DFSORT that the key is in positions 7-16 ascending. Since SORTED is specified, indicating that the records are already in order by the specified binary key, DFSORT will copy the SORTJNF2 records. Since NOSEQCK is specified, DFSORT will not check that the records are in order by the key. (Only use NOSEQCK if you know for sure that the records are in order by the key.)

The JOIN statement tells DFSORT that the joined records should include the unpaired F1 and F2 records as well as the paired F1/F2 records.

The REFORMAT statement defines the fields to be extracted for the joined records in the order in which they are to appear, and includes an indicator in the last position that will be set to '1' if the key is found only in the F1 file, '2' if the key is found only in the F2 file, or 'B' if the key is found in the F1 file and in the F2 file. FIELDS=(F1:1,14,F2:1,20,?) tells DFSORT to create the joined records as follows:

```
Joined Record Positions   Extracted from
-----------------------   -----------------
1-14                      F1 positions 1-14
15-34                     F2 positions 1-20
35                        Indicator of where key was found
```

The OPTION COPY statement tells DFSORT to copy the joined records. The OUTFIL statements use the indicator in position 35 to determine where to find the F1 or F2 fields in the joined records and where to write the fields (F1ONLY, F2ONLY or BOTH).

Conceptually, JOINKEYS application processing proceeds as follows:

- Subtask1 copies the SORTJNF1 (F1) records as directed by the JOINKEYS statement. As a result, it copies the unchanged SORTJNF1 records to the main task.
- Subtask2 copies the SORTJNF2 (F2) records as directed by the JOINKEYS statement. As a result, it copies the unchanged SORTJNF2 records to the main task.
- The main task joins the records passed from subtask1 and subtask2 as directed by the specified JOINKEYS, JOIN and REFORMAT statements, resulting in the following joined records (paired and unpaired):

```
Carrie    F101No    Carrie    F201B
David     F102                    1
Frank     F103                    1
Holly     F104Yes   Holly     F202B
              Yes   Karen     F2032
              No    Sri Hari  F2042
Vicky     F105Yes   Vicky     F205B
```

For F1 records without a match in F2 (for example, the F102 record), the indicator in position 35 has a '1'. For F2 records without a match in F1 (for example, the F203 record), the indicator in position 35 has a '2'. For F1 records with a match in F2 (for example, the F101 and F201 records), the indicator in position 35 has a 'B'.

- The first OUTFIL statement finds records with a '1' in position 35. These are the F1 records without a match in F2. The F1 field is in positions 1-14 of the joined record, so those positions are written to the F1ONLY file. Thus, F1ONLY contains these records:

```
David     F102
Frank     F103
```

- The second OUTFIL statement finds records with a '2' in position 35. These are the F2 records without a match in F1. The F2 field is in positions 15-34 of the joined record, so those positions are written to the F2ONLY file. Thus, F2ONLY contains these records:

```
Yes   Karen     F203
No    Sri Hari  F204
```

- The third OUTFIL statement finds records with 'B' in position 35. These are the F1 and F2 records with a match. The F1 field is in positions 1-14 of the joined record and the F2 field is in positions 15-34 of the joined record, so each joined record is split into those two records and written to the BOTH file. The shorter F1 record is padded with blanks on the right to the length of the F2 record. Thus, BOTH contains these records:

```
Carrie    F101
No    Carrie    F201
Holly     F104
Yes   Holly     F202
Vicky     F105
Yes   Vicky     F205
```

Note: If you only want one record per key in BOTH, you can have the BUILD for FNAMES=BOTH specify the positions for just that record. For example, BUILD=(1,14) for the F1 records or BUILD=(15,20) for the F2 records.

Example 6 - Paired and unpaired F1/F2 records (FILL method)

```
//JKE6  EXEC  PGM=SORT
//SYSOUT    DD  SYSOUT=*
//SORTJNF1 DD DSN=FIRST.FILE,DISP=SHR
//SORTJNF2 DD DSN=SECOND.FILE,DISP=SHR
//F1ONLY DD SYSOUT=*
//F2ONLY DD SYSOUT=*
//BOTH DD SYSOUT=*
//SYSIN    DD    *
* Control statements for JOINKEYS application
  JOINKEYS FILE=F1,FIELDS=(1,10,A),SORTED,NOSEQCK
  JOINKEYS FILE=F2,FIELDS=(7,10,A),SORTED,NOSEQCK
  JOIN UNPAIRED,F1,F2
  REFORMAT FIELDS=(F1:1,14,F2:1,20),FILL=C'$'
* Control statements for main task (joined records)
  OPTION COPY
  OUTFIL FNAMES=F1ONLY,INCLUDE=(15,1,CH,EQ,C'$'),
    BUILD=(1,14)
  OUTFIL FNAMES=F2ONLY,INCLUDE=(1,1,CH,EQ,C'$'),
    BUILD=(15,20)
  OUTFIL FNAMES=BOTH,INCLUDE=(15,1,CH,NE,C'$',AND,1,1,CH,NE,C'$'),
    BUILD=(1,14,/,15,20)
/*
```

This example illustrates an alternate way to create three output files; with paired F1/F2 records, unpaired F1 records and unpaired F2 records. It produces the same output as Example 5. Whereas Example 5 uses an indicator in one position to determine where the key was found, Example 6 uses a fill character in different positions to determine where the key was found. Another drawback of the Example 6 method is that you must use a FILL character that does not appear in either input record. The Explanation for Example 6 is the same as for Example 5 up to the REFORMAT statement and then it differs as follows:

The REFORMAT statement defines the fields to be extracted for the joined records in the order in which they are to appear. FIELDS=(F1:1,14,F2:1,20) tells DFSORT to create the joined records as follows:

```
Joined Record Positions   Extracted from
-----------------------   -----------------
1-14                      F1 positions 1-14
15-34                     F2 positions 1-20
```

FILL=C'$' tells DFSORT to use $ as the fill character for the F1 field in an unpaired F2 record and for the F2 field in an unpaired F1 record. We use the FILL character to identify the unpaired records from each file; when used for this purpose, the default of FILL=X'40' is usually not a good choice. You must select a FILL character that will not appear in either input file.

The OPTION COPY statement tells DFSORT to copy the joined records. The OUTFIL statements use the presence or absence of the $ fill character in certain positions to determine where to find the F1 or F2 fields in the joined records and where to write the fields (F1ONLY, F2ONLY or BOTH).

Conceptually, JOINKEYS application processing proceeds as follows:

- Subtask1 copies the SORTJNF1 (F1) records as directed by the JOINKEYS statement. As a result, it copies the unchanged SORTJNF1 records to the main task.
- Subtask2 copies the SORTJNF2 (F2) records as directed by the JOINKEYS statement. As a result, it copies the unchanged SORTJNF2 records to the main task.
- The main task joins the records passed from subtask1 and subtask2 as directed by the specified JOINKEYS, JOIN and REFORMAT statements, resulting in the following joined records (paired and unpaired):

```
Carrie    F101No    Carrie    F201
David     F102$$$$$$$$$$$$$$$$$$$$
Frank     F103$$$$$$$$$$$$$$$$$$$$
Holly     F104Yes   Holly     F202
$$$$$$$$$$$$$$Yes   Karen     F203
$$$$$$$$$$$$$$No    Sri Hari  F204
Vicky     F105Yes   Vicky     F205
```

For F1 records without a match in F2 (for example, the F102 record), the F2 field is filled with the FILL character. For F2 records without a match in F1 (for example, the F203 record), the F1 field is filled with the FILL character. For F1 records with a match in F2 (for example, the F101 and F201 records), no FILL characters are used.

- The first OUTFIL statement finds records with the FILL character in position 15. These are the F1 records without a match in F2. The F1 field is in positions 1-14 of the joined record, so those positions are written to the F1ONLY file. Thus, F1ONLY contains these records:

```
David     F102
Frank     F103
```

- The second OUTFIL statement finds records with the FILL character in position 1. These are the F2 records without a match in F1. The F2 field is in positions 15-34 of the joined record, so those positions are written to the F2ONLY file. Thus, F2ONLY contains these records:

```
Yes   Karen     F203
No    Sri Hari  F204
```

- The third OUTFIL statement finds records without the FILL character in position 1 or position 15. These are the F1 and F2 records with a match. The F1 field is in positions 1-14 of the joined record and the F2 field is in positions 15-34 of the joined record, so each joined record is split into those two records and written to the BOTH file. The shorter F1 record is padded with blanks on the right to the length of the F2 record. Thus, BOTH contains these records:

```
Carrie    F101
No    Carrie    F201
Holly     F104
Yes   Holly     F202
Vicky     F105
Yes   Vicky     F205
```

Note: If you only want one record per key in BOTH, you can have the BUILD for FNAMES=BOTH specify the positions for just that record. For example, BUILD=(1,14) for the F1 records or BUILD=(15,20) for the F2 records.

Using JOINKEYS with ICETOOL SORT and COPY

You can use a JOINKEYS application with a COPY or SORT operator of ICETOOL, but not with any of the other ICETOOL operators. A new JKFROM operand can be used instead of a FROM(indd) operand for the SORT and COPY operators to tell DFSORT you will provide a JOINKEYS statement with F1=ddname1 for the F1 file and a JOINKEYS statement with F2=ddname2 for the F2 file, as well as JOIN and REFORMAT statements as needed. With JKFROM, ICETOOL will use 'JOINKEYS' in the ICE606I message instead of the FROM ddname and will terminate if a JOINKEYS application is not specified correctly.

For multiple JOINKEYS applications within a single ICETOOL step:

- You can reference different JOINKEYS F1 and F2 input files for the different SORT and/or COPY operators by using F1=ddname and F2=ddname on the various JOINKEYS statements as appropriate

- You can reference different subtaskn message and control files for the different SORT and/or COPY operators by using TASKID=id on the various JOINKEYS statements as appropriate.

Notes:

1. If you want to sort the joined records, use a SORT operator with USING(xxxx) in TOOLIN and a SORT control statement along with the appropriate JOINKEYS, JOIN and REFORMAT statements in xxxxCNTL (main task).
2. If you want to copy the joined records, use a COPY operator with USING(yyyy) in TOOLIN and the appropriate JOINKEYS, JOIN and REFORMAT statements in yyyyCNTL (main task).
3. If the JOINKEYS statement for F1 specifies SORTED, a copy function will be used for subtask1 automatically. Otherwise, a sort function will be used for subtask1 automatically.
4. If the JOINKEYS statement for F2 specifies SORTED, a copy function will be used for subtask2 automatically. Otherwise, a sort function will be used for subtask2 automatically.

Here's an example of using one SORT operator for a simple JOINKEYS application.

```
//JKTL1 EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//JNA DD DSN=MY.INPUTA,DISP=SHR
//JNB DD DSN=MY.INPUTB,DISP=SHR
//OUT DD SYSOUT=*
//TOOLIN DD *
* SORT operator with JOINKEYS application.
SORT JKFROM TO(OUT) USING(CTL1)
/*
//CTL1CNTL DD *
* JOINKEYS application control statements for SORT operator.
  JOINKEYS F1=JNA,FIELDS=(5,4,A)
  JOINKEYS F2=JNB,FIELDS=(11,4,A),SORTED
  REFORMAT FIELDS=(F1:1,20,F2:5,15)
* Main task control statement for SORT operator
* (operates on joined records).
  OPTION EQUALS
  SORT FIELDS=(1,3,ZD,A)
/*
```

Here's an example of using multiple COPY operators for JOINKEYS applications that preprocess different input files in different ways:

```
//JKTL2 EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//IN1 DD DSN=MY.IN1,DISP=SHR
//IN2 DD DSN=MY.IN2,DISP=SHR
//IN3 DD DSN=MY.IN3,DISP=SHR
//OUT1 DD SYSOUT=*
//OUT2 DD SYSOUT=*
//TOOLIN DD *
* First COPY operator with JOINKEYS application.
COPY JKFROM TO(OUT1) USING(CTL1)
* Second COPY operator with JOINKEYS application.
COPY JKFROM USING(CTL2)
/*
//CTL1CNTL DD *
* JOINKEYS application control statements for first COPY operator.
  JOINKEYS F1=IN1,FIELDS=(5,12,A),TASKID=T1
  JOINKEYS F2=IN2,FIELDS=(11,12,A),TASKID=T1
  JOIN UNPAIRED
  REFORMAT FIELDS=(F1:4,40,F2:15,20),FILL=C'$'
* Main task control statements for first COPY operator
* (operates on joined records).
  INCLUDE COND=(8,1,CH,EQ,C'Y')
/*
//CTL2CNTL DD *
* JOINKEYS application control statements for second COPY operator.
  JOINKEYS F1=IN1,FIELDS=(5,12,A),TASKID=T1
  JOINKEYS F2=IN3,FIELDS=(9,12,A),TASKID=T2,SORTED
  REFORMAT FIELDS=(F1:4,40,F2:7,20)
* Main task control statements for second COPY operator
* (operates on joined records).
  OUTFIL FNAMES=OUT2,HEADER1=('Analysis Report'),REMOVECC
/*
//T1F1CNTL DD *
* Control statements for subtask1 (F1=IN1) of both COPY operators.
* Subtask1 sorts/joins on 5,12,A automatically
* per JOINKEYS statement for TASKID=T1/F1=IN1.
  INCLUDE COND=(21,3,CH,EQ,C'J82')
  SUM FIELDS=NONE
/*
//T1F2CNTL DD *
* Control statements for subtask2 (F2=IN2) of first COPY operator.
* Subtask1 sorts/joins on 11,12,A automatically
* per JOINKEYS statement for TASKID=T1/F2=IN2.
  OPTION SKIPREC=1
  INCLUDE COND=(25,3,CH,EQ,C'J82')
/*
//T2F2CNTL DD *
* Control statements for subtask2 (F2=IN3) of second COPY operator.
* Subtask1 copies/joins on 9,12,A automatically
* per JOINKEYS statement for TASKID=T2/F2=IN3/SORTED.
  INCLUDE COND=(5,3,CH,EQ,C'J82')
/*
```

Date Field Conversions

Introduction

You can use the BUILD or OVERLAY operands of the INREC, OUTREC and OUTFIL statements to convert input date fields of one type to corresponding output date fields of another type. You can convert date fields between combinations of 2-digit and 4-digit year dates, CH/ZD and PD dates, and Julian and Gregorian dates. You can also convert a date field to a corresponding day of the week in several forms.

Syntax

The syntax for the date field conversions in the BUILD and OVERLAY operands is as follows:

```
p,m,Yxx,todate
p,m,Yxx,WEEKDAY=CHAR3/CHAR9/DIGIT1
p,m,Yxx,DT=(abcd)
p,m,Yxx,DTNS=(abc)
```

%nn (a parsed field) can be used wherever p,m can be used.

DFSORT Symbols can be used for p,m,Y2x and p,m,Y4x fields.

Detailed Description for Date Field Conversions

p,m,Yxx,todate

Can be used to convert an input date field of one type to a corresponding output date field of another type.

Each type of date field you can use as input for date conversion is shown in Table 1.

Table 1 (Page 1 of 2). Input fields for p,m,Yxx conversion

m,Yxx	Input date
5,Y2T	C'yyddd' or Z'yyddd'
6,Y2T	C'yymmdd' or Z'yymmdd'
7,Y4T	C'ccyyddd' or Z'ccyyddd'
8,Y4T	C'ccyymmdd' or Z'ccyymmdd'
5,Y2W	C'dddyy' or Z'dddyy'
6,Y2W	C'mmddyy' or Z'mmddyy'
7,Y4W	C'dddccyy' or Z'dddccyy'
8,Y4W	C'mmddccyy' or Z'mmddccyy'
3,Y2U	P'yyddd'
4,Y2V	P'yymmdd'
4,Y4U	P'ccyyddd'
5,Y4V	P'ccyymmdd'
3,Y2X	P'dddyy'
4,Y2Y	P'mmddyy'

Table 1 (Page 2 of 2). Input fields for p,m,Yxx conversion

m,Yxx	**Input date**
4,Y4X	P'dddccyy'
5,Y4Y	P'mmddccyy'

todate can be one of the following:

- **TOJUL=Yaa**

  Converts the input date to a Julian output date.

- **TOJUL=Yaa(s)**

  Converts the input date to a Julian output date with s separators. s can be any character except a blank.

- **TOGREG=Yaa**

  Converts the input date to a Gregorian output date.

- **TOGREG=Yaa(s)**

  Converts the input date to a Gregorian output date with s separators. s can be any character except a blank.

The output date field created by each valid todate combination is shown in Table 2.

Table 2. TOJUL and TOGREG output date fields

Yaa	**TOJUL=Yaa**	**TOJUL=Yaa(s)**	**TOGREG=Yaa**	**TOGREG=Yaa(s)**
Y2T	C'yyddd'	C'yysddd'	C'yymmdd'	C'yysmmsdd'
Y2W	C'dddyy'	C'dddsyy'	C'mmddyy'	C'mmsddsyy'
Y2U	P'yyddd'	n/a	n/a	n/a
Y2V	n/a	n/a	P'yymmdd'	n/a
Y2X	P'dddyy'	n/a	n/a	n/a
Y2Y	n/a	n/a	P'mmddyy'	n/a
Y4T	C'ccyyddd'	C'ccyysddd'	C'ccyymmdd'	C'ccyysmmsdd'
Y4W	C'dddccyy'	C'dddsccyy'	C'mmddccyy'	C'mmsddsccyy'
Y4U	P'ccyyddd'	n/a	n/a	n/a
Y4V	n/a	n/a	P'ccyymmdd'	n/a
Y4X	P'dddccyy'	n/a	n/a	n/a
Y4Y	n/a	n/a	P'mmddccyy'	n/a

Example

```
* Convert a P'dddyy' input date to a C'ccyy/mm/dd' output date
  INREC BUILD=(21,3,Y2X,TOGREG=Y4T(/),X,
* Convert a C'ccyymmdd' input date to a P'ccyyddd' output date
     42,8,Y4T,TOJUL=Y4U,X,
* Convert a C'mmddyy' input date to a C'yymmdd' output date
     11,6,Y2W,TOGREG=Y2T)
```

p,m,Yxx,WEEKDAY=CHAR3/CHAR9/DIGIT1

Can be used to convert an input date field to a corresponding output day of the week in one of several forms.

Each type of date field you can use as input is shown in Table 1 on page 36.

The different types of output you can display are shown in Table 3.

Table 3. Output for weekdays

Day	**CHAR3**	**CHAR9**	**DIGIT1**
Sunday	C'SUN'	C'SUNDAY '	C'1'
Monday	C'MON'	C'MONDAY '	C'2'
Tuesday	C'TUE'	C'TUESDAY '	C'3'
Wednesday	C'WED'	C'WEDNESDAY'	C'4'
Thursday	C'THU'	C'THURSDAY '	C'5'
Friday	C'FRI'	C'FRIDAY '	C'6'
Saturday	C'SAT'	C'SATURDAY '	C'7'

Example

```
* Convert a P'mmddccyy' input date to a 3-character weekday
  OUTREC BUILD=(5:15,5,Y4Y,WEEKDAY=CHAR3,
* Convert a C'yyddd' input date to a 1-digit weekday
     18:27,5,Y2T,WEEKDAY=DIGIT1,
* Convert a P'dddccyy' input date to a 9-character weekday
     41:121,4,Y4X,WEEKDAY=CHAR9)
```

p,m,Yxx,DT=(abcd)

p,m,Yxx,DTNS=(abc)

Can be used to convert an input date field of one type to a corresponding Gregorian output date field of another type.

Each type of date field you can use as input is shown in Table 1 on page 36.

DT=(abcd) creates an output date in the form C'adbdc', where a, b, and c indicate the order in which the month, day, and year are to appear and whether the year is to appear as two or four digits, and d is the character to be used to separate the month, day and year. For a, b, and c, use M to represent the month (01-12), D to represent the day (01-31), Y to represent the last two digits of the year (for example, 09), or 4 to represent the four digits of the year (for example, 2009). M, D, and Y or 4 can each be specified only once.

DTNS=(abc) creates an output date in the form C'abc', where a, b and c indicate the order in which the month, day, and year are to appear and whether the year is to appear as two or four digits. For a, b and c, use M to represent the month (01-12), D to represent the day (01-31), Y to represent the last two digits of the year (for example, 09), or 4 to represent the four digits of the year (for example, 2009). M, D, and Y or 4 can each be specified only once.

Example

```
* Convert a C'yyddd' input date to a C'dd/mm/ccyy' output date
  OUTFIL BUILD=(92,5,Y2T,DT=(DM4/),X,
* Convert a P'ccyyddd' input date to a C'mmddyy' output date
     53:32,4,Y4U,DTNS=(MDY))
```

Conversion of Real Dates, Special Indicators and Invalid Dates

For CH/ZD dates (Y2T, Y4T, Y2W, Y4W), the zone and sign are ignored; only the digits are used. For example, X'F2F0F0F9F1F2F3D0' and X'A2B0C0D9E1F21320' are treated as 20091230. For PD dates (Y2U, Y4U, Y2V, Y4V, Y2X, Y4X, Y2Y, Y4Y), the sign is ignored; only the relevant digits are used. For example, X'120091230D' is treated as 20091230.

For CH/ZD dates (Y2T, Y4T, Y2W, Y4W), the special indicators are X'00...00' (BI zeros), X'40...40' (blanks), C'0...0' (CH zeros), Z'0...0' (ZD zeros), C'9...9' (CH nines), Z'9...9' (ZD nines) and X'FF...FF' (BI ones). For PD dates (Y2U, Y4U, Y2V, Y4V, Y2X, Y4X, Y2Y, Y4Y), the special indicators are P'0...0' (PD zeros) and P'9...9' (PD nines).

yy for real 2-digit year dates is transformed to ccyy when appropriate using the century window established by the Y2PAST option in effect. ccyy for real 4-digit year dates is transformed to yy when appropriate by removing cc.

Date conversion is not performed for special indicators; the special indicator is just used appropriately for the output date field. For example, if p,5,Y2T,TOGREG=Y4T(/) is used, an input yyddd special indicator of C'99999' results in an output date field of C'9999/99/99'. However, CH/ZD special indicators of BI zeros, blanks and BI ones cannot be converted to PD special indicators.

Conversion involving an input date with an invalid digit (A-F) will result in a data exception (0C7 ABEND) or an incorrect output value.

Conversion involving an invalid input date or invalid output date will result in an output value of asterisks and an informational message (issued once). A date is considered invalid if any of the following range conditions are not met:

- yy must be between 00 and 99
- ccyy must be between 0001 and 9999
- mm must be between 01 and 12
- dd must be between 01 and 31, and must be valid for the year and month
- ddd must be 001 to 366 for a leap year, or between 001 and 365 for a non-leap year.

A date is also considered invalid if the input field is a CH/ZD special indicator of binary zeros, blanks or binary ones, and the output field is PD.

Example 1 - Use of TOJUL, TOGREG and WEEKDAY

```
  OPTION COPY,Y2PAST=1996
  INREC BUILD=(1,6,Y2W,TOJUL=Y4T,X,
     1,6,Y2W,WEEKDAY=CHAR3,X,
     9,7,Y4T,TOGREG=Y4T(/),X,
     9,7,Y4T,WEEKDAY=DIGIT1)
```

This example illustrates how to convert an mmddyy date to a ccyyddd date and a 3-character weekday string, and how to convert a ccyyddd date to a ccyy/mm/dd date and 1-digit weekday string.

The input records might be as follows:

```
120409  1999014
051895  2003235
999999  0000000
013099  1992343
```

The output records would be as follows:

```
2009338 FRI 1999/01/14 5
2095138 WED 2003/08/23 7
9999999 999 0000/00/00 0
1999030 SAT 1992/12/08 3
```

The Y2PAST=1996 option sets the century window to 1996-2095. The century window is used to transform yy in the Y2W field to ccyy.

Note that date conversion is not performed for the special indicators (all 9s and all 0s); the special indicator is just used appropriately for the output date field.

Example 2 - Identifying Invalid Date Values

```
  SORT FIELDS=(1,12,CH,A)
  OUTREC OVERLAY=(30:16,8,Y4T,TOGREG=Y4T)
  OUTFIL INCLUDE=(30,1,CH,EQ,C'*')
```

This example illustrates how to list records with dates outside of the valid range (for example, a month not between 01-12).

Note: Dates with an invalid digit (A-F) can result in a data exception (0C7 ABEND).

The input records might be as follows:

```
Betten         20091021
Vezinaw        20091101
Casad          00000000
Boenig         20091325
Kolusu         20090931
Yaeger         20090731
```

The SORT statement sorts the records by the name in positions 1-12. After the SORT statement is processed, the sorted records will look like this:

```
Betten         20091021
Boenig         20091325
Casad          00000000
Kolusu         20090931
Vezinaw        20091101
Yaeger         20090731
```

The OUTREC statement uses TOGREG to convert each ccyymmdd value in positions 16-23 to a ccyymmdd value in positions 30-37; the third column will be identical to the second column for valid dates and special indicators, but will contain asterisks for invalid dates. After the OUTREC statement is processed, the reformatted records will look like this:

```
Betten        20091021      20091021
Boenig        20091325      ********
Casad         00000000      00000000
Kolusu        20090931      ********
Vezinaw       20091101      20091101
Yaeger        20090731      20090731
```

The OUTFIL statement selects the records that have an asterisk in position 30, that is, the records with an invalid date. The Boenig record is invalid because mm is 13, and the Kolusu record is invalid because mm is 09 but dd is 31 (September only has 30 days). Note that the special indicator of all 0s for the Casad record is valid. The output records would be as follows:

```
Boenig        20091325      ********
Kolusu        20090931      ********
```

Date Field Editing

You can use the BUILD or OVERLAY operands of the INREC, OUTREC and OUTFIL statements to edit and convert 4-digit year date fields in several ways previously available for 2-digit year date fields.

The syntax for 4-digit year date field editing in the BUILD and OVERLAY operands is as follows:

```
p,m,Y4x<,edit/to>
p,m,Y4x(s)
```

%nn (a parsed field) can be used wherever p,m can be used.

DFSORT Symbols can be used for p,m,Y4x fields.

p,m,Y4x<,edit/to>

Can be used to convert a 4-digit year input date field to an output field using the edit or to items specified. Each type of date field you can use as input and the resulting field for p,m,4x is shown in Table 4.

Table 4. Input and result fields for Y4x date editing

m,Y4x	Input date	Result field for p,m,Y4x
7,Y4T	C'ccyyddd' or Z'ccyyddd'	C'ccyyddd'
8,Y4T	C'ccyymmdd' or Z'ccyymmdd'	C'ccyymmdd'
7,Y4W	C'dddccyy' or Z'dddccyy'	C'dddccyy'
8,Y4W	C'mmddccyy' or Z'mmddccyy'	C'mmddccyy'
4,Y4U	P'ccyyddd'	C'ccyyddd'
5,Y4V	P'ccyymmdd'	C'ccyymmdd'
4,Y4X	P'dddccyy'	C'dddccyy'
5,Y4Y	P'mmddccyy'	C'mmddccyy'

If edit or to items are used, they are applied to the result field shown in Table 4.

The edit items are those described for p,m,f,edit in *z/OS DFSORT Application Programming Guide* (M0-M26, EDIT, SIGNS, LENGTH).

The to items are those described for p,m,f,to in *z/OS DFSORT Application Programming Guide* (TO,LENGTH).

Example

```
* Convert a C'mmddccyy' date to a C'mmddccyy' date.
  OUTFIL BUILD=(34,8,Y4W,X,
* Convert a P'ccyymmdd' date to a C'ccyy-mm-dd' date.
      13,5,Y4V,EDIT=(TTTT-TT-TT),X,
* Convert a C'dddccyy' date to a 4-byte BI dddccyy value.
      61,7,Y4W,TO=BI,LENGTH=4)
```

p,m,Y4x(s)

Can be used to convert a 4-digit year input date field to an output date with separators. s can be any character except a blank. Each type of date field you can use as input and the resulting field for p,m,4x(s) is shown in Table 5.

Table 5. Input and result fields for Y4x(s) date editing

m,Y4x	Input date	Result field for p,m,Y4x(s)
7,Y4T	C'ccyyddd' or Z'ccyyddd'	C'ccyysddd'
8,Y4T	C'ccyymmdd' or Z'ccyymmdd'	C'ccyysmmsdd'
7,Y4W	C'dddccyy' or Z'dddccyy'	C'dddsccyy'
8,Y4W	C'mmddccyy' or Z'mmddccyy'	C'mmsddsccyy'
4,Y4U	P'ccyyddd'	C'ccyysddd'
5,Y4V	P'ccyymmdd'	C'ccyysmmsdd'
4,Y4X	P'dddccyy'	C'dddsccyy'
5,Y4Y	P'mmddccyy'	C'mmsddsccyy'

Example

```
* Convert a Z'dddccyy' date to a C'ddd/ccyy' date.
  OUTFIL BUILD=(19,7,Y4W(/),X,
* Convert a P'ccyymmdd' date to a C'ccyy-mm-dd' date.
      43,5,Y4V(-))
```

Editing of Special Indicators and Invalid Dates

For CH/ZD dates (Y4T, Y4W), the special indicators are X'00...00' (BI zeros), X'40...40' (blanks), C'0...0' (CH zeros), Z'0...0' (ZD zeros), C'9...9' (CH nines), Z'9...9' (ZD nines) and X'FF...FF' (BI ones). For PD dates (Y4U, Y4V, Y4X, Y4Y), the special indicators are P'0...0' (PD zeros) and P'9...9' (PD nines).

A special indicator is just edited appropriately for the output date field. For example, if p,7,Y4T(/) is used, an input ccyyddd special indicator of C'99999999' results in an output date field of C'9999/99/99'.

Editing involving an input date with an invalid digit (A-F) can result in a data exception (0C7 ABEND) or an incorrect output value.

Editing involving an invalid input date can result in an invalid output value.

Example 1 - Use of Y4x(s)

```
  OPTION COPY
  OUTFIL BUILD=(1,4,Y4X(-))
```

This example illustrates how a P'dddccyy' date can be edited to a C'ccyy-ddd' date. For example, if an input record has a PD value of 2342008 in positions 1-4, the output record will have a character value of 2008-234 in positions 1-8.

MERGE Operator

Introduction

MERGE is a new ICETOOL operator that allows you to merge up to 50 input data sets that are already in order by the same keys. Various options of MERGE allow you to define the ddnames for the input and output data sets and the MERGE and other DFSORT control statements to be used for the MERGE operation.

As an example, you could use the following ICETOOL step to merge four input data sets to a single output data set.

```
//MRG1 EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//DFSMSG  DD SYSOUT=*
//IN1 DD DSN=MY.INPUT1,DISP=SHR
//IN2 DD DSN=MY.INPUT2,DISP=SHR
//IN3 DD DSN=MY.INPUT3,DISP=SHR
//IN4 DD DSN=MY.INPUT4,DISP=SHR
//OUT DD DSN=MY.OUTPUT1,DISP=(NEW,CATLG,DELETE),
// UNIT=SYSDA,SPACE=(CYL,(5,5))
//TOOLIN DD *
  MERGE FROM(IN1,IN2,IN3,IN4) TO(OUT) USING(CTL1)
//CTL1CNTL DD *
  MERGE FIELDS=(11,8,ZD,A)
```

Syntax

The syntax for the MERGE operator is as follows:

```
MERGE FROM(indd<,indd>...) <FROM(indd<,indd>...)> ... USING(xxxx)
  <TO(outdd<,outdd>...)>  <VSAMTYPE(x)>
  <LOCALE(name/CURRENT/NONE)>  <SERIAL>
```

Detailed Description

Merges up to 50 input data sets to an output data set. The records in each input data set to be merged must already be in sorted order as specified by the control fields in a supplied DFSORT MERGE statement.

You must specify at least one FROM operand and a USING(xxxx) operand. Each FROM operand can be used to specify one or more ddnames. You can specify up to 10 FROM operands. The maximum number of ddnames in all of the FROM operands must not exceed 50.

DFSORT is called to merge the indd data sets to the outdd data sets using the DFSORT control statements in xxxxCNTL. You must supply a DFSORT MERGE statement in the xxxxCNTL data set to indicate the control fields for the merge. You can use additional DFSORT statements in the xxxxCNTL data set to merge a subset of the input records (INCLUDE or OMIT statement; OUTFIL INCLUDE, OMIT, SAVE, STARTREC, ENDREC, SAMPLE, SPLIT, SPLITBY and SPLIT1R operands), reformat records for output (INREC, OUTREC, and OUTFIL statements; user exit routines), and so on.

If EQUALS is in effect, records that collate identically are output in the order of their ddnames in the FROM operands.

When ICETOOL is called using the parameter list interface, the 1-byte operation status indicator in the Return Area will be set to 0 or 4 for a MERGE operator in the same way as for existing operators. No operation specific values are returned for MERGE.

The active locale's collating rules affect MERGE, INCLUDE and OMIT processing as explained in *z/OS DFSORT Application Programming Guide*.

Note: For a merge application, records deleted during an E35 exit routine are not sequence checked. If you use an E35 exit routine without an output data set, sequence checking is not performed at the time the records are passed to the E35 user exit; therefore, you must ensure that input records are in correct sequence.

The operands described below can be specified in any order:

- **FROM(indd,...)**

  Specifies the ddnames of the input data sets to be read by DFSORT for this operation. Up to 10 FROM operands can be used to specify up to 50 input ddnames. An indd DD statement must be present for each indd name specified. Each indd input data set must conform to the rules for DFSORT's SORTINnn data sets.

- **USING(xxxx)**

  Specifies the first 4 characters of the ddname for the control statement data set to be used by DFSORT for this operation. xxxx must be four characters that are valid in a ddname of the form xxxxCNTL. xxxx must not be SYSx.

  An xxxxCNTL DD statement must be present, and the control statements in it must conform to the rules for DFSORT's SORTCNTL data set.

  The xxxxCNTL data set must contain a MERGE statement. If TO is not specified, the xxxxCNTL data set must also contain either one or more OUTFIL statements or a MODS statement for an E35 routine that disposes of all records. Other statements are optional.

- **TO(outdd,...)**

  Specifies the ddnames of the output data sets to be written by DFSORT for this operation. From 1 to 10 outdd names can be specified. An outdd DD statement must be present for each outdd name specified. If a single outdd data set is specified, DFSORT is called once to merge the indd data sets to the outdd data set using SORTOUT processing; the outdd data set must conform to the rules for DFSORT's SORTOUT data set. If multiple outdd data sets are specified and SERIAL is not specified, DFSORT is called once to merge the indd data sets to the outdd data sets using OUTFIL processing; the outdd data sets must conform to the rules for DFSORT's OUTFIL data sets.

  A ddname specified in a FROM operand must not also be specified in the TO operand.

- **VSAMTYPE(x)**

  Specifies the record type for a VSAM input data set. x must be either F for fixed-length record processing or V for variable-length record processing.

If VSAMTYPE(x) is specified, ICETOOL will pass a RECORD TYPE=x control statement to DFSORT. (If you specify a RECORD TYPE=x statement in the xxxxCNTL data set, it will override the one passed by ICETOOL.)

- **LOCALE(name)**

  Specifies that locale processing is to be used and designates the name of the locale to be made active during DFSORT processing. LOCALE(name) can be used to override the LOCALE installation option. For complete details on LOCALE(name), see the discussion of the LOCALE operand in *z/OS DFSORT Application Programming Guide*.

- **LOCALE(CURRENT)**

  Specifies that locale processing is to be used, and the current locale active when DFSORT is entered will remain the active locale during DFSORT processing. LOCALE(CURRENT) can be used to override the LOCALE installation option. For complete details on LOCALE(name), see the discussion of the LOCALE operand in *z/OS DFSORT Application Programming Guide*.

- **LOCALE(NONE)**

  Specifies that locale processing is not to be used. DFSORT will use the binary encoding of the code page defined for your data for collating and comparing. LOCALE(NONE) can be used to override the LOCALE installation option. For complete details on LOCALE(name), see the discussion of the LOCALE operand in *z/OS DFSORT Application Programming Guide*.

- **SERIAL**

  Specifies that OUTFIL processing is not to be used when multiple outdd data sets are specified. DFSORT is called multiple times and uses SORTOUT processing; the outdd data sets must conform to the rules for DFSORT's SORTOUT data set. SERIAL is not recommended because the use of serial processing (that is, multiple calls to DFSORT) instead of OUTFIL processing can degrade performance and imposes certain restrictions as detailed below. SERIAL is ignored if a single outdd data set is specified.

  DFSORT is called to merge the indd data set to the first outdd data set using the DFSORT control statements in the xxxxCNTL data set. If the merge operation is successful, DFSORT is called as many times as necessary to copy the first outdd data set to the second and subsequent outdd data sets. Therefore, for maximum efficiency, use a disk data set as the first in a list of outdd data sets on both disk and tape. If more than one outdd data set is specified, DFSORT must be able to read the first outdd data set after it is written in order to copy it to the other outdd data sets. Do not use a SYSOUT or DUMMY data set as the first in a list of outdd data sets because:

  – If the first data set is SYSOUT, DFSORT abends when it tries to copy the SYSOUT data set to the second outdd data set.

  – If the first data set is DUMMY, DFSORT copies the empty DUMMY data set to the other outdd data sets (that is, all of the resulting outdd data sets are empty).

Example 1 - MERGE five input files to one output file

```
//TOOLIN DD *
MERGE FROM(IN01,IN02,IN03,IN04,IN05) TO(OUTPUT) USING(MERG)
//MERGCNTL DD *
  OPTION EQUALS
  MERGE FIELDS=(21,4,CH,A)
/*
```

This example merges 5 input files to an output file. EQUALS is used to ensure that records that collate identically are output in the order specified in the FROM operand. For example, if IN01, IN03 and IN05 all have records with a key or 'AAAA' in positions 21-24, the output will contain the 'AAAA' record from IN01, the 'AAAA' record from IN03 and the 'AAAA' record from IN05, in that order.

Example 2 - Merge seven input files to two output files

```
//TOOLIN DD *
MERGE FROM(INPUT1,INPUT2,INPUT3,INPUT4) -
      FROM(INPUT5,INPUT6,INPUT7) VSAMTYPE(F) USING(MRG1)
//MRG1CNTL DD *
  MERGE FIELDS=(52,8,UFF,D)
  OUTFIL FNAMES=OUT1,INCLUDE=(15,3,SS,EQ,C'D21,D33')
  OUTFIL FNAMES=OUT2,SAVE
/*
```

This example merges 7 input files to 2 output files. It uses two OUTFIL statements to create the two output files; each output file will have a different subset of the merged records. VSAMTYPE(F) tells DFSORT the record type is F (only needed for VSAM input files).

MERGEIN alternate ddnames

Introduction

A new MERGEIN operand can be specified on an OPTION statement to supply alternate ddnames for the input data sets used for a MERGE application. If MERGEIN is passed in DFSPARM or an extended parameter list, the specified ddnames will be used instead of the default SORTINnn ddnames. Up to 100 ddnames can be specified with MERGEIN.

Note: If MERGEIN is passed in SYSIN or SORTCNTL, the specified ddnames will not be used (instead, SORTINnn ddnames will be used).

Syntax

The syntax for MERGEIN is as follows:

```
  OPTION MERGEIN=(ddname<,ddname>...)
```

Detailed Description

Specifies up to 100 ddnames to be be used for a MERGE application instead of the SORTINnn ddnames. This allows you to use any valid alternate ddnames for the MERGE data sets.

If EQUALS is in effect, records that collate identically are output in the order of their ddnames in the MERGEIN list.

Each ddname can be 1 through 8 characters. If a ddname is specified more than once in the MERGEIN operand, it will only be used once. If more than 100 unique ddnames are specified in the MERGEIN operand, only the first 100 will be used. Do not use ddnames reserved for use by DFSORT, such as SYSOUT, ccccOUT, ccccWKd, ccccWKdd, ccccDKd, or ccccDKdd, where cccc is the specified or defaulted value for the SORTDD operand and d is any character. Do not use the same ddnames for MERGEIN and OUTFIL.

Note: MERGEIN is processed only if it is passed on the OPTION control statement in an extended parameter list, or in DFSPARM.

Default: SORTINnn, unless SORTDD=cccc is specified in an extended parameter list or in DFSPARM, in which case ccccINnn is the default.

Example 1 - Use of three alternate ddnames for MERGE

```
//S1 EXEC  PGM=SORT
//SYSOUT    DD  SYSOUT=*
//SARA DD *
AAA FROM SARA
CCC FROM SARA
DDD FROM SARA
//MOLLY DD *
AAA FROM MOLLY
BBB FROM MOLLY
DDD FROM MOLLY
//NORA DD *
AAA FROM NORA
BBB FROM NORA
CCC FROM NORA
//SORTOUT DD SYSOUT=*
//DFSPARM  DD    *
 OPTION EQUALS,MERGEIN=(NORA,SARA,MOLLY)
 MERGE FIELDS=(1,3,CH,A)
/*
```

This example illustrates the use of the alternate ddnames NORA, SARA and MOLLY for a MERGE application instead of SORTINnn ddnames. Since EQUALS is specified, equally collating records will be from NORA, then SARA, then MOLLY, that is, in the order specified in the MERGEIN list. Thus, SORTOUT contains these records:

```
AAA FROM NORA
AAA FROM SARA
AAA FROM MOLLY
BBB FROM NORA
BBB FROM MOLLY
CCC FROM NORA
CCC FROM SARA
DDD FROM SARA
DDD FROM MOLLY
```

If MERGEIN=(SARA,MOLLY,NORA) was specified in the MERGEIN list in DFSPARM, SORTOUT would contain these records:

```
AAA FROM SARA
AAA FROM MOLLY
AAA FROM NORA
BBB FROM MOLLY
BBB FROM NORA
CCC FROM SARA
CCC FROM NORA
DDD FROM SARA
DDD FROM MOLLY
```

New Messages

This section shows messages that have been added for PTFs UK51706 and UK51707. Refer to *z/OS DFSORT Messages, Codes and Diagnosis Guide* for general information on DFSORT messages.

ICE288I

ICE288I INPUT OR OUTPUT DATE VALUE OUT OF RANGE FOR DATE CONVERSION

Explanation: For a date conversion operation using TOJUL, TOGREG, WEEKDAY, DT or DTNS, an invalid input date was used.

A date value is considered invalid if any of the following range conditions are not met:

- yy must be between 00 and 99
- ccyy must be between 0001 and 9999
- mm must be between 01 and 12
- dd must be between 01 and 31, and must be valid for the year and month
- ddd must be 001 to 366 for a leap year, or between 001 and 365 for a non-leap year.

A date is also considered invalid if the input field is a CH/ZD special indicator of binary zeros, blanks or binary ones, and the output field is PD.

System Action: Asterisks are printed for each invalid output value. The message is only issued once. Processing continues.

Programmer Response: Check for output values containing asterisks and ensure that the input date value is valid and that you are not converting a CH/ZD special indicator of binary zeros, blanks or binary ones to a PD value.

ICE400A

ICE400A INVALID JOINKEYS, JOIN OR REFORMAT STATEMENT OPERAND

Explanation: Critical. An invalid keyword operand was detected on a JOINKEYS, JOIN or REFORMAT control statement.

System Action: The program terminates.

Programmer Response: Make sure the JOINKEYS, JOIN or REFORMAT control statement contains only valid keyword operands.

ICE401A

ICE401A DUPLICATE JOINKEYS, JOIN OR REFORMAT STATEMENT OPERAND

Explanation: Critical. One of the following errors was found:

- On a JOINKEYS, JOIN or REFORMAT statement, a keyword was specified twice.
- On a JOINKEYS statement, more than one of FILE=F1, FILES=F1, F1=ddname, FILE=F2, FILES=F2 or F2=ddname was specified.
- On a JOINKEYS statement, INCLUDE and OMIT were both specified.

System Action: The program terminates.

Programmer Response: Check the JOINKEYS, JOIN or REFORMAT control statement for the errors indicated in the explanation and correct the error.

ICE402A

ICE402A JOINKEYS STATEMENT FOR Fn WAS REQUIRED, BUT NOT FOUND

Explanation: Critical. A JOINKEYS application was requested by a JOINKEYS, JOIN or REFORMAT statement, or a JKFROM operand (ICETOOL COPY or SORT), but a JOINKEYS statement for file F1 or file F2, as indicated, was not found.

System Action: The program terminates.

Programmer Response: Supply two JOINKEYS statements; one for F1 (with FILE=F1, FILES=F1 or F1=ddname) and another for F2 (with FILE=F2, FILES=F2 or F2=ddname).

ICE403A

ICE403A OPERAND keyword WAS REQUIRED FOR verb STATEMENT, BUT NOT FOUND

Explanation: Critical. The indicated required keyword was missing for the indicated control statement as follows:

- For a JOINKEYS statement, FILE=F1, FILES=F1, FILE=F2, FILES=F2, F1=ddname or F2=ddname must be specified.
- For a JOINKEYS statement, FIELDS must be specified.
- For a JOIN statement, UNPAIRED must be specified.
- For a REFORMAT statement, FIELDS must be specified.

System Action: The program terminates.

Programmer Response: Specify the indicated keyword for the indicated control statement.

ICE404A

ICE404A REFORMAT STATEMENT WAS REQUIRED, BUT NOT FOUND

Explanation: Critical. A JOIN statement with an ONLY operand was not specified, so a REFORMAT statement is required. However, a REFORMAT statement was not found.

System Action: The program terminates.

Programmer Response: Either specify a JOIN statement with the ONLY operand, or specify a REFORMAT statement, as appropriate.

ICE405A

ICE405A JOINKEYS STATEMENTS HAD MISMATCH IN NUMBER, LENGTH OR ORDER OF KEYS

Explanation: Critical. The keys (p,m,s) specified in the FIELDS operands of the JOINKEYS statements for F1 and F2 did not match in one or more of the following ways:

- The two FIELDS operands have different numbers of keys. For example, FIELDS for F1 has two keys and FIELDS for F2 has three keys.
- Corresponding keys in the two FIELDS operands have different lengths. For example, the second key for F1 has a length of 5 and the second key for F2 has a length of 6.
- Corresponding keys in the two FIELDS operands have different orders. For example, the third key for F1 has ascending order (A) and the third key for F2 has descending order (D).

System Action: The program terminates.

Programmer Response: Ensure that the FIELDS operands of the JOINKEYS statements for F1 and F2 have the same number of keys, and that corresponding keys have the same length and order.

ICE406A

ICE406A JOINKEYS STATEMENT FIELD ENDS AFTER POSITION 32752

Explanation: Critical. The last byte of the key (p,m,s) in a JOINKEYS FIELDS operand ended beyond position 32752. Each key must end at or before position 32752 (position plus length must not be greater than 32753). For example, 32752,1,A is valid because it ends at position 32752, but 32752,2,A is invalid because it ends at position 32753.

System Action: The program terminates.

Programmer Response: Ensure that each key in the JOINKEYS FIELDS operand ends at or before position 32752.

ICE407A

ICE407A JOINKEYS STATEMENT HAD TOTAL KEY LENGTH GREATER THAN 4080 BYTES

Explanation: Critical. The total length of all of the keys (p,m,s) in a JOINKEYS FIELDS operand exceeded the limit of 4080 bytes.

System Action: The program terminates.

Programmer Response: Ensure that the total length of all of the keys in the JOINKEYS FIELDS operand is less than or equal to 4080 bytes.

ICE408A

ICE408A MERGE FUNCTION CANNOT BE USED WITH JOINKEYS MAIN TASK

Explanation: Critical. A MERGE FIELDS=(p,m,s,...) statement was found for the main task of a JOINKEYS application in SYSIN, SORTCNTL, DFSPARM or a parameter list. A MERGE function cannot be used with JOINKEYS.

System Action: The program terminates.

Programmer Response: Replace the MERGE FIELDS=(p,m,s,...) statement with a MERGE FIELDS=COPY, SORT FIELDS=COPY, OPTION COPY or SORT FIELDS=(p,m,s,...) statement, as appropriate.

ICE409A

ICE409A INSUFFICIENT STORAGE FOR JOINKEYS APPLICATION - ADD AT LEAST nMB

Explanation: Critical. DFSORT could not get the additional nMB of storage needed for this JOINKEYS application.

System Action: The program terminates.

Programmer Response: Add at least nMB to the storage available to DFSORT (for example, increase the REGION size).

ICE410A

ICE410A JOINKEYS APPLICATION TERMINATED - SEE ddname MESSAGES

Explanation: Critical. This message is issued by subtask1 (for file F1) or subtask2 (for file F2) of a JOINKEYS application to indicate that the main task terminated. ddname is the ddname associated with the message data set for the main task.

System Action: The program terminates.

Programmer Response: See the messages in the indicated ddname data set for information about the main task. Correct the error that caused the main task to terminate.

ICE411I

ICE411I THIS IS THE JOINKEYS MAIN TASK FOR JOINING F1 AND F2

Explanation: Indicates this is the main task for a JOINKEYS application. The main task processes the joined records from input files F1 and F2 and writes the output.

System Action: None.

Programmer Response: None.

ICE412A

ICE412A REFORMAT REQUIRES RDW IN FIRST FIELD

Explanation: Critical. The FIELDS operand of the REFORMAT statement has a position without a length (p without m) as its last field, but the first field does not include the RDW (1,n with n equal to or greater than 4). For example, the REFORMAT statement is:

```
  REFORMAT FIELDS=(F1:5,8,F2:1,20,F1:15)
```

instead of:

```
  REFORMAT FIELDS=(F1:1,8,F2:1,20,F1:15)
```

or the REFORMAT statement is:

```
  REFORMAT FIELDS=(?,F1:1,4,5)
```

instead of:

```
  REFORMAT FIELDS=(F1:1,4,?,F1:5)
```

System Action: The program terminates.

Programmer Response: Include the RDW (1,n with n equal to or greater than 4) in the first field, or do not use a position without a length, as appropriate.

ICE413A

ICE413A REFORMAT REQUIRES TYPE=V FILE FOR RDW AND VARIABLE FIELDS

Explanation: Critical. The FIELDS operand of the REFORMAT statement has a position without a length (p without m) as its last field, but has one of the following errors:

- F1: is used for the first field (which includes the RDW), but F1 does not refer to a TYPE=V file.
- F2: is used for the first field (which includes the RDW), but F2 does not refer to a TYPE=V file.
- F1: is used for a position without length field, but F1 does not refer to a TYPE=V file.
- F2: is used for a position without length field, but F2 does not refer to a TYPE=V file.

For example, SORTJNF1 for F1 has RECFM=FB and SORTJNF2 for F2 has RECFM=VB and the REFORMAT statement is:

```
  REFORMAT FIELDS=(F1:1,20,F2:5,6,15)
```

instead of:

```
  REFORMAT FIELDS=(F2:1,4,F1:1,20,F2:5,6,15)
```

System Action: The program terminates.

Programmer Response: Ensure that the file (F1 or F2) used for the first field, and for each position without length field, is a TYPE=V file.

ICE414A

ICE414A ddname (Fn) type FIELD END AT p IS BEYOND LENGTH OF n

Explanation: Critical. A JOINKEYS or REFORMAT statement specifies a field which ends beyond the maximum record length. The information displayed in the message is as follows:

- the ddname of the input file with the field in error.
- the file (F1 or F2) with the field in error.
- the type of field in error as follows:
  - KEY to indicate a field in the FIELDS operand of a JOINKEYS statement.
  - INCLUDE to indicate a field in the INCLUDE operand of a JOINKEYS statement.
  - OMIT to indicate a field in the OMIT operand of a JOINKEYS statement.
  - REFORMAT to indicate a field in the FIELDS operand of a REFORMAT statement.
- the position (p) at which the field ends.
- the maximum record length (n) which the ending position exceeded.

System Action: The program terminates.

Programmer Response: Ensure that all fields specified in JOINKEYS and REFORMAT statements are contained within the maximum record length indicated.

ICE415A

ICE415A TYPE=x JOINED RECORD LENGTH OF n EXCEEDS MAXIMUM OF m

Explanation: Critical. The maximum length (n) of the joined records for a JOINKEYS application, as defined by the REFORMAT statement or by default if a REFORMAT statement was not specified, exceeds the maximum length (m) of 32760 for TYPE=F records or 32767 for TYPE=V records.

System Action: The program terminates.

Programmer Response: Reduce the maximum length of the joined records so it does not exceed the maximum length of 32760 for TYPE=F records or 32767 for TYPE=V records.

Note that the maximum LRECL for RECFM=VB records is 32756 and the maximum LRECL for RECFM=VBS records is 32767; if you want RECFM=VBS records, ensure that RECFM is specified appropriately for the output data set.

ICE416I

ICE416I JOINKEYS IS USING THE Fn SUBTASK FOR ddname1 - SEE ddname2 MESSAGES

Explanation: Indicates DFSORT is using a subtask to process the Fn (F1 or F2) file for a JOINKEYS application. ddname1 is the ddname associated with the input file for the subtask. ddname2 is the ddname associated with the message data set for the subtask.

System Action: None.

Programmer Response: See the messages in the indicated ddname2 data set for information about the subtask.

ICE417I

ICE417I THIS IS THE JOINKEYS Fn SUBTASK FOR ddname

Explanation: Indicates this is a subtask used to process the Fn (F1 or F2) file for a JOINKEYS application. The subtask passes the needed Fn fields to the main task. ddname is the ddname for the input file associated with the subtask.

System Action: None.

Programmer Response: None.

ICE418A

ICE418A JOINKEYS Fn SUBTASK FOR ddname1 TERMINATED - SEE ddname2 MESSAGES

Explanation: Critical. This message is issued by the main task of a JOINKEYS application to indicate that the subtask used to process file Fn (F1 or F2) terminated. ddname1 is the ddname associated with the input file for the subtask. ddname2 is the ddname associated with the message data set for the subtask.

System Action: The program terminates.

Programmer Response: See the messages in the ddname2 data set for information about the subtask. Correct the error that caused the subtask to terminate.

ICE419I

ICE419I JOINED RECORDS: TYPE=x, LENGTH=n

Explanation: Indicates the record type (F or V) and maximum record length (n) of the joined records passed as input to the main task of a JOINKEYS application.

System Action: None.

Programmer Response: None.

ICE420A

ICE420A COULD NOT ALLOCATE ddname FOR Fn MESSAGES - SUPPLY DD STATEMENT

Explanation: Critical. For a JOINKEYS application, a required message data set was not found and could not be dynamically allocated. A DD statement for the indicated ddname was required for the messages associated with the subtask for the F1 or F2 file. A DD statement for that ddname was not found, so DFSORT attempted to dynamically allocate a SYSOUT=* message data set. However, the message data set could not be allocated.

System Action: The program terminates.

Programmer Response: Supply a message data set using a DD statement for the indicated ddname.

ICE421I

ICE421I JOINED RECORDS: COUNT=n

Explanation: Indicates the number (n) of joined records passed as input to the main task of a JOINKEYS application.

System Action: None.

Programmer Response: None.

ICE422I

ICE422I JOINKEYS STATEMENT FOR Fn FOUND PREVIOUSLY - THIS STATEMENT IGNORED

Explanation: A JOINKEYS statement for the indicated file (F1 or F2) was found previously in this source (for example, SYSIN) or in a higher source (for example, DFSPARM is higher than SYSIN). The second and subsequent JOINKEYS statement for F1 or F2 in the same source or in a lower source is ignored.

System Action: None.

Programmer Response: Correct the conflicting JOINKEYS statements, if appropriate.

ICE423A

ICE423A REFORMAT STATEMENT FIELD ENDS AFTER POSITION 32767

Explanation: Critical. The last byte of a field (p,m) in a REFORMAT FIELDS operand ended beyond position 32767. Each field must end at or before position 32767 (position plus length must not be greater than 32768). For example, 32767,1 is valid because it ends at position 32767, but 32767,2 is invalid because it ends at position 32768.

System Action: The program terminates.

Programmer Response: Ensure that each field in the REFORMAT FIELDS operand ends at or before position 32767.

ICE424A

ICE424A ddname (Fn) KEY IS OUT OF SEQUENCE

Explanation: Critical. The SORTED operand was specified without the NOSEQCK operand on the JOINKEYS statement associated with the indicated ddname for file F1 or F2. DFSORT checked the records of the indicated file and found a record out of sequence for the keys specified in the FIELDS operand of the JOINKEYS statement.

System Action: The program terminates.

Programmer Response: Remove the SORTED operand from the JOINKEYS statement for the indicated file (F1 or F2) to force DFSORT to sort that file by the specified keys.

ICE425A

ICE425A ddname CANNOT BE USED AS DDNAME FOR BOTH F1 AND F2

Explanation: Critical. The indicated ddname was used on the JOINKEYS statements for both file F1 and F2 (for example, one JOINKEYS statement has F1=IN1 and the other JOINKEYS statement has F2=IN1). Different ddnames must be used for the two files.

System Action: The program terminates.

Programmer Response: Change the F1, F2 or FILE operand in one or both of the JOINKEYS statements to use different ddnames for the two files.

ICE426A

ICE426A cccc CANNOT BE USED AS PREFIX FOR JOINKEYS MAIN TASK AND SUBTASK

Explanation: Critical. For a JOINKEYS application, the SORTDD value for the main task and the TASKID value for a subtask resulted in the same four character prefix indicated by cccc (for example, SORTDD=MYF1 and TASKID=MY both result in a prefix of MYF1). Different prefixes must be used for the main task and the subtasks.

System Action: The program terminates.

Programmer Response: Change the SORTDD operand for the main task, or the TASKID operand for the subtask, to use different prefixes.

ICE427A

ICE427A verb STATEMENT CANNOT BE USED WITH JOINKEYS SUBTASK

Explanation: Critical. For a JOINKEYS application, a JOINKEYS, JOIN, MERGE, OUTFIL, OUTFILE, OUTREC, REFORMAT or SORT statement was found in JNF1CNTL for subtask1 or in JNF2CNTL for subtask2. These statements cannot be used for a JOINKEYS subtask.

System Action: The program terminates.

Programmer Response: Remove all statements that cannot be used for a JOINKEYS subtask from JNF1CNTL or JNF2CNTL.

ICE428A

ICE428A TOO MANY DUPLICATES OF ONE KEY IN ddname (F2)

Explanation: Critical. The F2 file associated with the indicated ddname contained more duplicates for a single key than DFSORT could process with the storage available.

System Action: The program terminates.

Programmer Response: If possible, increase the storage available to DFSORT (for example, try specifying REGION=0M). Alternatively, if the maximum duplicates for a single key in file F1 is less than the maximum duplicates for a single key in file F2, reverse F1 and F2. For example, if you received this error with these control statements:

```
JOINKEYS F1=IN1,FIELDS=(11,4,A)
JOINKEYS F2=IN2,FIELDS=(21,4,A)
REFORMAT FIELDS=(F1:1,20,F2:5,30)
```

and the maximum duplicates for a single F1 key is less than the maximum duplicates for a single F2 key, use these control statements instead:

```
JOINKEYS F1=IN2,FIELDS=(21,4,A)
JOINKEYS F2=IN1,FIELDS=(11,4,A)
REFORMAT FIELDS=(F2:1,20,F1:5,30)
```

ICE429A

ICE429A JOINKEYS APPLICATION IS ONLY ALLOWED WITH SORT OR COPY OPERATOR

Explanation: Critical. A JOINKEYS, JOIN or REFORMAT statement was specified for an ICETOOL operator other than SORT or COPY. A JOINKEYS application is only allowed for a SORT or COPY operator, not for any of the other operators.

System Action: The program terminates.

Programmer Response: Redesign the application to not use JOINKEYS, JOIN or REFORMAT with ICETOOL operators other than SORT or COPY.

ICE657A

ICE657A TOO MANY FROM DDNAMES FOR MERGE

Explanation: Critical. The maximum of 50 FROM ddnames was exceeded for this MERGE operation.

System Action: The operation is terminated.

Programmer Response: A $ marks the point at which the error was detected. Reduce the number of FROM ddnames for this MERGE operator to 50 or less. Use additional MERGE operators to handle all of the data sets required.

ICE658A

ICE658A MERGE FUNCTION IS REQUIRED FOR MERGE OPERATION

Explanation: Critical. A DFSORT MERGE statement was not found for this MERGE operator, but you must supply a MERGE statement with a MERGE operator.

System Action: The operation is terminated.

Programmer Response: Specify a MERGE statement in the xxxxCNTL data set corresponding to the USING(xxxx) operand for this MERGE operator. Ensure that the MERGE statement is not overridden by a SORT statement, an OPTION COPY statement, or a MERGE FIELDS=COPY statement.

Changed Messages

This section shows existing messages that have been changed significantly for PTFs UK51706 and UK51707. Refer to *z/OS DFSORT Messages, Codes and Diagnosis Guide* for general information on DFSORT messages.

ICE005A

JOINKEYS, JOIN and REFORMAT are added to the list of operation definers.

ICE018A, ICE113A, ICE114A

These messages will be issued for the INCLUDE and OMIT operands of a JOINKEYS statement for the same reasons they are issued for the INCLUDE and OMIT operands of an OUTFIL statement.

ICE022A

This message will be issued for the TYPE operand of a JOINKEYS statement for the same reasons it is issued for the TYPE operand of a RECORD statement.

ICE039A

For a JOINKEYS application, try using REGION=0M.

ICE054I

For a JOINKEYS subtask, 0 is always displayed for OUT. The number of joined records is displayed in message ICE421I for the main task.

ICE056A

This message will be issued for a JOINKEYS operation when a DD statement identified in the FILE=F1, FILES=F1, F1=ddname, FILE=F2, FILES=F2 or F2=ddname operand of a JOINKEYS statement is not supplied. The F1 ddname is SORTJNF1 (or ccccJNF1 if SORTDD=cccc is in effect) if FILE=F1 or FILES=F1 is specified. The F2 ddname is SORTJNF2 (or ccccJNF2 if SORTDD=cccc is in effect) if FILE=F2 or FILES=F2 is specified.

This message can also be issued when a ddname specified in a MERGEIN operand is not supplied or is the same as the SORTOUT ddname.

ICE068A

ICE068A OUT OF SEQUENCE {ddname|SORTINnn}

OUT OF SEQ SORTINnn

Explanation: Critical. During a merge, a data set was found to be out of sequence.

If Blockset was selected:

- If input was not supplied through exit E32, ddname is the ddname associated with the data set which was found to be out of sequence
- If input was supplied through user exit E32, SORTINnn identifies the file which was found to be out of sequence. 00 signifies the first input file, 01 the second, and so on.

If Blockset was not selected:

- If input was not supplied through exit E32, SORTINnn is the ddname associated with the data set which was found to be out of sequence
- If input was supplied through user exit E32, SORTINnn identifies the file which was found to be out of sequence. 01 signifies the first input file, 02 the second, and so on.

System Action: The operation is terminated.

Programmer Response: If a user-written routine was modifying the records, check the routine thoroughly. It should not modify control fields at user exit E35. If a user-written routine is not being used, make sure that all input data sets have been sorted on the same control fields, in the same order, and that they all have a similar format. Check whether you have also received message ICE072I.

If input was supplied through user exit E32, check your routine to make sure records are passed to the merge from the correct file.

If you were reading in variable-length VSAM records through user exit E32, check the format and accuracy of the record descriptor word (RDW) you built at the beginning of each record.

If Blockset was not selected, 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.

ICE083A, ICE098I, ICE118I, ICE253I, ICE254I, ICE258I, ICE298I

If the information in any of these messages indicates you should specify AVGRLEN=n, FILSZ=En, DYNSPC=n or DYNALLOC=(,n) for a JOINKEYS operation, you can supply the needed parameters as follows:

- In DFSPARM for the JOINKEYS main task.
- In JNF1CNTL for the JOINKEYS F1 subtask.
- In JNF2CNTL for the JOINKEYS F2 subtask.

ICE111A

This message will be issued for the following additional situations:

- DT=(abcd) or DTNS=(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.
- The length (m for p,m or FIXLEN=m for %nn) for a Y4 format field was not 7-8 for Y4T or Y4W, 4 for Y4U or Y4X, or 5 for for Y4V or Y4Y.

ICE151A, ICE221A

These messages will be issued for the INCLUDE and OMIT operands of a JOINKEYS statement for the same reasons they are issued for the COND operand of an INCLUDE or OMIT statement.

ICE189A

This message will be issued for the following additional situations if Blockset could not be used:

- JOINKEYS operation
- Y4x format
- MERGEIN processing

ICE217A

The appropriate MERGE ddname is now displayed instead of 'SORTINNN'.

ICE218A

This message can be issued in the JOINKEYS F1 or F2 subtask for an INCLUDE or OMIT operand of a JOINKEYS statement for the same reason it is issued for the COND operand of an INCLUDE or OMIT statement. Note that the INCLUDE or OMIT operand of the JOINKEYS statement will be displayed in the main task, not in the subtask. You can use the DFSPARM data set to specify VLSCMP if appropriate.

If you specify an INCLUDE or OMIT statement in the JNFnCNTL data set for the JOINKEYS F1 or F2 subtask instead of an INCLUDE or OMIT operand in the JOINKEYS statement, the INCLUDE or OMIT statement will be displayed for the subtask, and you can specify VLSCMP in the JNFnCNTL data set if appropriate.

ICE272A

Y4T, Y4U, Y4V, Y4W, Y4X and Y4Y and mixed and lowercase variations are added to the list of valid formats (f) for p,m,f.

ICE276A

Y4x (uppercase Y only), where x is any character, is added to the list of reserved words for symbols.

ICE606I

For a MERGE operator, 'MERGE' will be displayed as the function and 'MERGEIN' will be displayed for ddname1.

For a COPY or SORT operator, if JKFROM is specified, 'JOINKEYS' will be displayed for ddname1.

ICE613A

The required keywords for a MERGE operator are FROM and USING.

The required keywords for a COPY operator are now FROM and TO or USING, or JKFROM and USING.

The required keywords for a SORT operator are now FROM or JKFROM, and USING.

ICE614A

MERGE is now a valid operator.

ICE623A

The maximum number of FROM operands for a MERGE operator is 10.

ICE624A

The maximum number of TO ddnames for a MERGE operator is 10.