DFSORT PTF UQ90053 User Guide

Document Number GC27-2000-00

March, 2004

Frank L. Yaeger

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

DFSORT/MVS Web Site

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

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

Abstract

This paper is the documentation for DFSORT PTF **UQ90053**, which was first made available for DFSORT Release 14 in **February, 2003**. This PTF was also incorporated into z/OS DFSORT V1R5. UQ90053 provides important enhancements for join and match operations, sampling, repeating and distributing records, arithmetic operations using numeric fields and decimal constants, longer fields for substring searches, easier migration from other sort products, and more. This paper highlights, describes, and shows examples of the new features provided by this PTF for DFSORT and for DFSORT's powerful, multi-purpose ICETOOL utility. It also details changed messages associated with this PTF.

Contents

DFSORT PTF UQ90053 User Guide

Introduction

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

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

DFSORT Release 14 PTF **UQ90053**, first available in **February, 2003** and incorporated into z/OS DFSORT V1R5, provides important enhancements for join and match operations, sampling, repeating and distributing records, arithmetic operations using numeric fields and decimal constants, longer fields for substring searches, easier migration from other sort products, and more.

This paper highlights, describes, and shows examples of the new features provided by PTF UQ90053 for DFSORT and for DFSORT's powerful, multi-purpose ICETOOL utility. It also details changed messages associated with this PTF.

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 PTF UQ90053. The information in this paper is included in the z/OS DFSORT V1R5 books, but not in the DFSORT Release 14 books.

You should refer to *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 *DFSORT Messages, Codes and Diagnosis Guide* for general information on DFSORT messages.

Summary of Enhancements

ICETOOL Enhancements

A new SPLICE operator helps you to perform various file "join" and "match" operations. SPLICE allows you to create output records in a variety of ways by splicing together fields from records that have the same key, but different information. For example, for the same userid, you could create output records consisting of the division and department from one type of record (originating in input file1) with the account number and phone number from another type of record (originating in input file2). Spliced records can be created by combining the first duplicate and:

- one or more fields from the last duplicate, or
- one or more fields from each subsequent duplicate, or
- one field from each subsequent duplicate.

Non-duplicate records can be deleted or kept.

The USING(xxxx) option can now be used with ICETOOL's SELECT operator to process DFSORT control statements like INCLUDE, OMIT and OUTFIL for a SELECT operation.

New LISTSDB and LISTNOSDB options for ICETOOL's DEFAULTS, DISPLAY and OCCUR operators allow you to control the use of system-determined optimum blocksize for LIST data sets.

OUTFIL Enhancements

New SAMPLE=n and SAMPLE=(n,m) options of OUTFIL allow you to sample records in a variety of ways.

A new REPEAT=n option of OUTFIL allows you to write each output record multiple times.

A new SPLITBY=n option of OUTFIL allows you to write groups of records in rotation among multiple output data sets.

OUTFIL OUTREC now allows you to insert decimal constants (+n and -n) in your records as BI, FI, PD, ZD, FS or edited CH values.

OUTFIL OUTREC now allows you to combine fields (p,m,f), decimal constants (+n and -n), operators (MIN, MAX, MUL, DIV, MOD, ADD, SUB) and parentheses to form arithmetic expressions, and place the results in your records as BI, FI, PD, ZD, FS or edited CH values.

A new DATE4 option of OUTFIL OUTREC allows you to insert a timestamp for your DFSORT run in the form 'yyyy-mm-dd-hh.mm.ss' into your records.

The maximum length for an SS field used with OUTFIL INCLUDE and OUTFIL OMIT has been raised to 32752.

A new DATE4 option of OUTFIL INCLUDE and OUTFIL OMIT allows you to compare fields to a timestamp for your DFSORT run in the form 'yyyy-mm-dd-hh.mm.ss' or to a portion of that timestamp truncated on the right.

A PD0 field can now be compared to a hexadecimal constant or to another PD0 field for OUTFIL INCLUDE and OMIT.

OUTFILE can now be used as an alias for OUTFIL.

INREC and OUTREC Enhancements

INREC and OUTREC now allow you to insert decimal constants (+n and -n) in your records as BI, FI, PD, ZD, FS or edited CH values.

INREC and OUTREC now allow you to combine fields (p,m,f), decimal constants (+n and -n), operators (MIN, MAX, MUL, DIV, MOD, ADD, SUB) and parentheses to form arithmetic expressions, and place the results in your records as BI, FI, PD, ZD, FS or edited CH values.

A new DATE4 option of INREC and OUTREC allows you to insert a timestamp for your DFSORT run in the form 'yyyy-mm-dd-hh.mm.ss' into your records.

INCLUDE and OMIT Enhancements

The maximum length for an SS field used with INCLUDE and OMIT has been raised to 32752.

A new DATE4 option of INCLUDE and OMIT allows you to compare fields to a timestamp for your DFSORT run in the form 'yyyy-mm-dd-hh.mm.ss' or to a portion of that timestamp truncated on the right.

A PD0 field can now be compared to a hexadecimal constant or to another PD0 field for INCLUDE and OMIT.

FORMAT=f can now be used with mixed p,m and p,m,f fields in the COND operand for INCLUDE and OMIT. f from FORMAT=f will be used for p,m fields but not for p,m,f fields.

SORT, MERGE and SUM Enhancement

FORMAT=f can now be used with mixed p,m and p,m,f fields in the FIELDS operand for SORT, MERGE and SUM. f from FORMAT=f will be used for p,m fields but not for p,m,f fields.

Other Enhancements

Enhancements to DFSORT's control statement continuation rules allow you to continue a line that breaks at column 71 anywhere in columns 2 to 16 of the next line.

When PARMDDN=ddname is specified at installation-time, DFSORT will now use a //DFSPARM DD data set if a //ddname DD data set is not present. When PARMDDN=DFSPARM is specified or defaulted at installation-time, DFSORT will continue to use a //$ORTPARM DD data set if a //DFSPARM DD data set is not present.

ICETOOL Enhancements

This section discusses the following new features of DFSORT's powerful, multi-purpose ICETOOL utility:

- USING(xxxx) operand for the SELECT operator
- LISTSDB and LISTNOSDB operands for the DEFAULTS, DISPLAY and OCCUR operators
- SPLICE operator

Refer to Chapter 6 of *DFSORT Application Programming Guide* for general information on DFSORT's ICETOOL, and in particular for the framework of existing ICETOOL features upon which these new features are built.

USING with SELECT

The USING(xxxx) operand can now be used with ICETOOL's SELECT operator to process DFSORT control statements like INCLUDE, OMIT and OUTFIL for a SELECT operation. USING(xxxx) is optional for SELECT. The TO(outdd) operand and/or the DISCARD(savedd) operand must be specified even if the USING(xxxx) operand is specified.

General Description

The DFSORT control statements in xxxxCNTL are used if USING(xxxx) is specified with SELECT. You can use DFSORT control statements and options in the xxxxCNTL data set such as INCLUDE, OMIT, OPTION and OUTFIL to eliminate records, reformat records, and so on.

When ICETOOL calls DFSORT, it passes control statements and options appropriate for the SELECT operation being performed. To avoid unintended results or abends, you should not use USING(xxxx) and xxxxCNTL to override the DFSORT control statements or options passed by ICETOOL unless you understand the ramifications of doing so. In particular:

- Do not supply your own DFSORT INREC, MODS, OUTREC or SORT statement.
- You can use an INCLUDE or OMIT statement to remove input records **before** SELECT processing.

- If you specify TO(outdd) without DISCARD(savedd), you can further process the outdd records **after** SELECT processing using one (and only one) OUTFIL statement like this:

  ```
  OUTFIL FNAMES=outdd,...
  ```

- If you specify DISCARD(savedd) without TO(outdd), you can further process the savedd records **after** SELECT processing using one (and only one) OUTFIL statement like this:

  ```
  OUTFIL FNAMES=savedd,...
  ```

- If you specify TO(outdd) and DISCARD(savedd), you can further process the outdd and savedd records **after** SELECT processing using two (and only two) OUTFIL statements like this:

  ```
  OUTFIL FNAMES=outdd,...
  OUTFIL FNAMES=savedd,...
  ```

  Both statements must be specified in the order shown with at least the FNAMES option. For example, to further modify only the DISCARD data set, you could use statements like this:

  ```
  OUTFIL FNAMES=OUT
  OUTFIL FNAMES=SAVE,INCLUDE=(21,3,ZD,GT,+25)
  ```

The DYNALLOC option is passed to DFSORT to ensure that work space is available for the sort. If your installation defaults for dynamic allocation are inappropriate for a SELECT operator, you can specify USING(xxxx) and take one of the following actions:

1. Override the DYNALLOC option using an OPTION control statement such as:

   ```
   OPTION DYNALLOC=(,8)
   ```

   in the xxxxCNTL data set.

2. Use xxxxWKdd DD statements to override the use of dynamic allocation . See "SORTWKdd DD Statement" in Chapter 2 of *DFSORT Application Programming Guide* for details.

Operand Description

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 which are valid in a ddname of the form xxxxCNTL. xxxx must not be SYSx.

If USING is specified, an xxxxCNTL DD statement must be present and the control statements in it:

1. Must conform to the rules for DFSORT's SORTCNTL data set.
2. Should generally be used only for an INCLUDE or OMIT statement, comment statements, or appropriate OUTFIL statements as described under “General Description” on page 3.

SELECT Example with USING

```
+-------------------------------------------------------------------+
| ...                                                               |
| //TOOLIN DD *                                                     |
|    SELECT FROM(INPUT) TO(ONLYONE) ON(23,3,FS) NODUPS USING(CTL1)  |
| /*                                                                |
| //CTL1CNTL DD *                                                   |
|   OUTFIL FNAMES=ONLYONE,                                          |
|     REMOVECC,                                                     |
|     INCLUDE=(23,3,FS,LT,100),                                     |
|     OUTREC=(1:1,7,8:C'|',11:11,7,19:C'|',23:23,3,FS,M11,          |
|            27:C'|',30:30,15),                                     |
|     TRAILER1=(/,'TOTAL= ',TOT=(23,3,FS,M11,LENGTH=6))             |
| /*                                                                |
+-------------------------------------------------------------------+
```

This SELECT example shows how you can use USING(xxxx) to supply an OUTFIL statement to modify the TO data set.

The SELECT operator sorts the INPUT data set, selecting only the records from INPUT with floating sign values in positions 23-25 that occur just once (that is, only records with no duplicate ON field values).

The OUTFIL statement in CTL1CNTL is used to further modify the selected records for output to the ONLYONE data set.

The ONLYONE data set might look as follows:

```
DFSRT2 |  EISSLER |   005 |  DOC.EXAMPLES
DFSRT1 |  PACKER  |   008 |  ICETOOL.SMF.RUN
USR002 |  EISSLER |   012 |  DOC.EXAMPLES
SYS003 |  YAEGER  |   032 |  ICETOOL.TEST.CA

TOTAL= 000057
```

LISTSDB and LISTNOSDB

New LISTSDB and LISTNOSDB operands for ICETOOL's DEFAULTS, DISPLAY and OCCUR operators allow you to control the use of system-determined optimum blocksize for LIST data sets. LISTSDB and LISTNOSDB are optional, and mutually exclusive, for DEFAULTS, DISPLAY and OCCUR.

General Description

ICETOOL's DEFAULTS, DISPLAY and OCCUR operators each produce output in a list data set associated with a LIST(listdd) operand. If the BLKSIZE for the list data set is not available and the system-determined optimum block size can be used, the BLKSIZE will be set as directed by the new LISTSDB or LISTNOSDB operand if specified, or otherwise as directed by the SDBMSG installation option from ICEAM2 or ICEAM4.

Note: LISTSDB has no effect for SYSOUT list data sets (for example, //RPT1 DD SYSOUT=*), since the system-determined optimum block size is not used for spool or dummy data sets. See the description of the SDBMSG option in *DFSORT Installation and Customization* for more information on when the system-determined optimum block size can be used for list data sets.

Operand Description

LISTSDB or LISTNOSDB

Can be used to override the SDBMSG value for this LIST data set. LISTSDB directs ICETOOL to select the system-determined optimum block size for the LIST data set in the same way as for installation option

SDBMSG=YES. LISTNOSDB directs ICETOOL to select the block size for the LIST data set in the same way as for installation option SDBMSG=NO.

DISPLAY Example with LISTSDB

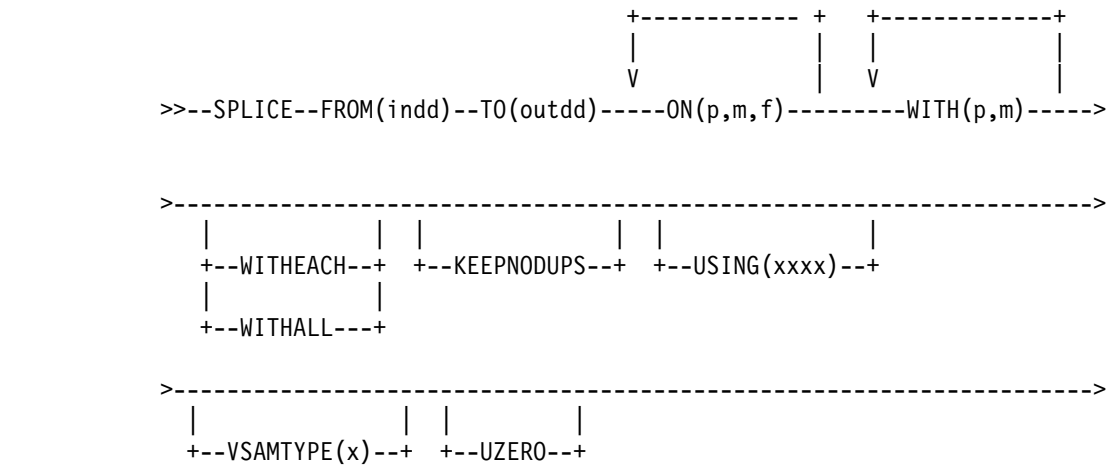
```
+-------------------------------------------------------------+
| ...                                                         |
| //RPT1 DD DSN=&O1,UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(,PASS)  |
| //TOOLIN DD *                                               |
|   DISPLAY FROM(IN) LIST(RPT1) BLANK DATE ON(VLEN) LISTSDB   |
| /*                                                          |
+-------------------------------------------------------------+
```

The LISTSDB operand ensures that the BLKSIZE for the RPT1 list data set is set to the system-determined optimum block size.

SPLICE Operator

SPLICE is a new ICETOOL operator that helps you to perform various file "join" and "match" operations. SPLICE allows you to create output records in a variety of ways by splicing together fields from records that have the same key, but different information.

Syntax



General Description

Splices together specified fields from records with matching numeric or character field values (that is, duplicate values), but different information. This makes it possible to join fields from different types of input records to create an output record with information from two or more records.

Typically, you will want to reformat the records from two or more data sets to temporary data sets, and concatenate those temporary data sets together as input to the SPLICE operator (be sure to follow the rules for concatenated data sets discussed in Chapter 2 of *DFSORT Application Programming Guide*).

"SPLICE Examples" on page 10 shows some techniques for splicing records from different data sets together in a variety of ways to perform various file "join" and "match" operations.

By default (when WITHALL and WITHEACH are not specified), one spliced record is created for each set of duplicates. The first duplicate is spliced with specified fields from the last duplicate.

The first duplicate is treated as a "base record". The last duplicate is treated as an "overlay record". Specified fields from the overlay record are overlaid on to the base record. Thus, the output record consists of fields from the base (first) record intermixed with specified fields from the overlay (last) record.

The records to be spliced can originate from two different input data sets.

From 1 to 10 ON fields can be used for the fields to match on. At least one ON(p,m,f) field must be specified; all such ON fields specified will be used to the determine the matching records to be spliced together.

From 1 to 50 WITH fields can be used to specify the fields to be overlaid on the base record from the overlay record. At least one WITH(p,m) field must be specified; all such WITH fields specified will be overlaid on to the base record. All other fields in the base record will be kept unchanged.

To illustrate the splicing process, if we had the following two input records with the base fields, ON field and WITH fields as shown:

```
BASE1  ON1    BASE2           BASE3 BASE4  GGGGG
       ON1            WITHA                WITHB
```

the resulting spliced output record would be:

```
BASE1  ON1    BASE2   WITHA   BASE3 BASE4  WITHB
```

WITHEACH can be used to create one spliced record for each set of duplicates. The first duplicate is spliced with one specified field from each subsequent duplicate.

The first duplicate is treated as a base record. Each subsequent duplicate is treated as an overlay record. The specified field from each overlay record is overlaid on to the base record. Thus, the output record consists of fields from the base record intermixed with a specified field from each overlay record. Note that the specified "field" from an overlay record can actually consist of multiple fields from the record that have previously been reformatted into one contiguous field.

The records to be spliced can originate from multiple input data sets.

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

```
BASE1  ON1     BASE2
       ON1             WITHA
       ON1                     WITHB
       ON1                             WITHC
```

The resulting spliced output record would be:

```
BASE1  ON1     BASE2  WITHA   WITHB  WITHC
```

WITHALL can be used to create multiple spliced records for each set of duplicates. The first duplicate is spliced with the specified fields from the second duplicate. Then the first duplicate is spliced with the specified fields from the third duplicate, and so on.

The first duplicate is treated as a base record. Each subsequent duplicate is treated as an overlay record. The specified fields from each overlay record are overlaid on to the base record. Thus, the output records consist of fields from the base record intermixed with specified fields from the overlay records.

The records to be spliced can originate from multiple input data sets.

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

```
BASE1  ON1    BASE2           BASE3 BASE4  GGGGG
       ON1            WITHA                WITHB
       ON1            WITHC
       ON1            WITHE                WITHF
```

The resulting three spliced output records would be:

```
BASE1  ON1    BASE2   WITHA   BASE3 BASE4  WITHB
BASE1  ON1    BASE2   WITHC   BASE3 BASE4
BASE1  ON1    BASE2   WITHE   BASE3 BASE4  WITHF
```

Note that without WITHALL, the resulting single spliced output record would be:

```
BASE1  ON1    BASE2   WITHE   BASE3 BASE4  WITHF
```

KEEPNODUPS can be used to keep the non-duplicate records as well as the spliced records. The non-duplicate records will be unchanged.

DFSORT is called to sort the indd data set. ICETOOL uses its E35 exit to determine which records to splice and include in the outdd data set. ICETOOL passes the EQUALS operand to DFSORT to ensure that duplicates are kept in their original input order.

The DFSORT control statements in xxxxCNTL are used if USING(xxxx) is specified. You can use DFSORT control statements and options in the xxxxCNTL data set such as INCLUDE, OMIT, OPTION and OUTFIL to eliminate records, reformat records, create reports, and so on.

When ICETOOL calls DFSORT, it passes control statements and options appropriate for the SPLICE operation being performed. To avoid unintended results or abends, you should not use USING(xxxx) and xxxxCNTL to override the DFSORT control statements or options passed by ICETOOL unless you understand the ramifications of doing so. In particular:

- Do not supply your own DFSORT INREC, MODS, OUTREC or SORT statement.
- You can use an INCLUDE or OMIT statement to remove input records **before** SPLICE processing.
- You can further process the outdd records associated with TO(outdd) **after** SPLICE processing using an OUTFIL statement like this:

  ```
  OUTFIL FNAMES=outdd,...
  ```

  For example, with TO(OUT1) you could further modify the OUT1 records after they have been spliced, with a statement like this:

  ```
  OUTFIL FNAMES=OUT1,FTOV,VLTRIM=X'40'
  ```

The DYNALLOC option is passed to DFSORT to ensure that work space is available for the sort. If your installation defaults for dynamic allocation are inappropriate for a SPLICE operator, you can specify USING(xxxx) and take one of the following actions:

1. Override the DYNALLOC option using an OPTION control statement such as:

   ```
   OPTION DYNALLOC=(,8)
   ```

   in the xxxxCNTL data set.

2. Use xxxxWKdd DD statements to override the use of dynamic allocation. See "SORTWKdd DD Statement" in Chapter 2 of *DFSORT Application Programming Guide* for details.

Tape work data sets **cannot** be used with ICETOOL.

Operand Descriptions

FROM(indd)
See the discussion of FROM(indd) for the COPY operator in Chapter 6 of *DFSORT Application Programming Guide*.

TO(outdd)
Specifies the ddname of the output data set to which DFSORT will write the records it produces for the operation (that is, the spliced records, and the non-duplicate records if KEEPNODUPS is specified).

An outdd DD statement must be present and must define an output data set that conforms to the rules for DFSORT's SORTOUT data set.

The ddname specified in the TO operand must not be the same as the ddname specified in the FROM operand.

See "JCL Restrictions" in Chapter 6 of *DFSORT Application Programming Guide* for more information.

ON(p,m,f)
See the discussion of ON(p,m,f) for the SELECT operator in Chapter 6 of *DFSORT Application Programming Guide*.

As with the other ICETOOL operators, you can use a symbol for an ON field. A symbol for p,m,f results in substitution of p,m,f if ON(symbol) is specified. A symbol for p,m or p,m,f results in substitution of p,m if ON(symbol,f) is specified.

WITH(p,m)
Specifies the position and length of a field to be overlaid from the overlay record on to the base record.

p specifies the first byte of the field relative to the beginning of the overlay record. p is 1 for the first **data** byte of a fixed-length record and 5 for the first **data** byte of a variable-length record as illustrated below (RRRR represents the 4-byte record descriptor word):

```
     Fixed-length record    |    Variable-length record
     | D | A | T | A | ...  |    | R | R | R | R | D | A | T | A | ...
  p=   1   2   3   4         | p=   1   2   3   4   5   6   7   8
```

m specifies the length of the field in bytes. A field must not extend beyond position 32752.

You can use a symbol for a WITH field. A symbol for p,m or p,m,f results in substitution of p,m if WITH(symbol) is specified.

A WITH field will not be used to overlay the RDW of a variable-length base record, to overlay bytes beyond the end of a base record, or to overlay bytes from beyond the end of an overlay record on to a base record. When necessary, WITH fields will be adjusted to prevent these situations. For example, if WITH(1,6) is specified for a variable-length record, it will be treated as WITH(5,2) and if WITH(75,10) is specified for an 80-byte base record or overlay record, it will be treated as WITH(75,6).

WITHEACH
Specifies that the first duplicate is spliced with one specified field from each subsequent duplicate. One WITH field from each overlay record is overlaid on to the base record. The first WITH field specifies the bytes to be overlaid from the second duplicate record on to the first duplicate record. The second WITH field specifies the bytes to be overlaid from the third duplicate record on to the first duplicate record, and so on. For any set of duplicates, extra overlay records without matching WITH fields, or extra WITH fields without matching overlay records are ignored.

With WITHEACH, a single spliced output record is created using the base record and one field from each overlay record. Note that the specified "field" from an overlay record can actually consist of multiple fields from the record that have previously been reformatted into one contiguous field.

WITHEACH overrides the default of splicing the first duplicate with all of the specified fields from the last duplicate.

WITHALL
Specifies that the first duplicate is spliced with specified fields from the second duplicate, and then from each subsequent duplicate in turn. All of the WITH fields from each overlay record are overlaid on to the base record.

With WITHALL, a spliced output record is created from each base base record and overlay record, resulting in n-1 spliced records for each set of n duplicates.

WITHALL overrides the default of splicing the first duplicate with all of the specified fields from the last duplicate.

KEEPNODUPS
Specifies that non-duplicates records are to be kept as well as spliced records. The non-duplicate records will be unchanged.

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 which are valid in a ddname of the form xxxxCNTL. xxxx must not be SYSx.

If USING is specified, an xxxxCNTL DD statement must be present and the control statements in it:

1. Must conform to the rules for DFSORT's SORTCNTL data set.
2. Should generally be used only for an INCLUDE or OMIT statement, comment statements, or appropriate OUTFIL statements as described under “General Description” on page 6.

VSAMTYPE(x)
See the discussion of VSAMTYPE(x) for the COPY operator in Chapter 6 of *DFSORT Application Programming Guide*.

UZERO
See the discussion of UZERO for the OCCUR operator in Chapter 6 of *DFSORT Application Programming Guide*.

SPLICE Examples

This section contains the following examples, discussed in detail, which illustrate just a few of the ways in which the SPLICE operator can be used, and the techniques involved:

1. Create one spliced record for each match in two files
2. Combine complete records from four files
3. Create multiple spliced records for each match in two files
4. Pull records from a master file in sorted order
5. Pull records from a master file in their original order
6. Create a report showing if needed parts are on-hand
7. Create a report showing if needed parts are on-hand - advanced

SPLICE normally requires reformatting the records of two or more data sets so they can be joined, so complete JCL examples are shown in this section to illustrate the suggested techniques. These techniques and others can be employed with SPLICE to perform a variety of tasks.

Since SPLICE overlays the WITH fields from the overlay record to the base record using matching ON fields, it's usually necessary to do some initial setup before using SPLICE, to ensure that:

- the ON fields are in the same positions in the base and overlay records
- the WITH fields in the overlay records are in the positions they will occupy in the base records
- the base records and overlay records are the same length.

For optimum efficiency, it is also a good idea to remove any records that are not needed for the SPLICE operation as part of the initial setup before the SPLICE operation, by using appropriate INCLUDE or OMIT statements.

Example 1 - Create one spliced record for each match in two files

This example shows how you can splice data together for each pair of records with the same ON field in two different input data sets.

```
+-------------------------------------------------------------------------+
| //S1    EXEC  PGM=ICETOOL                                               |
| //TOOLMSG   DD  SYSOUT=*                                                |
| //DFSMSG    DD  SYSOUT=*                                                |
| //IN1 DD *                                                              |
| Y12 89503 MKT                                                           |
| Y12 57301 MKT                                                           |
| Z35 02316 DEV                                                           |
| Y12 91073 MKT                                                           |
| Z35 18693 DEV                                                           |
| /*                                                                      |
| //IN2 DD *                                                              |
| 89503 27M $9,185,354 SAN JOSE   CA                                      |
| 72135 08M   $317,632 BOSTON     MA                                      |
| 18693 10M $8,732,105 BUFFALO    NY                                      |
| 57301 50M    $30,000 NEWARK     NJ                                      |
| /*                                                                      |
| //TEMP1 DD DSN=&&TEMP1,DISP=(,PASS),SPACE=(TRK,(5,5)),UNIT=SYSDA        |
| //TEMP2  DD DSN=&&TEMP2,DISP=(,PASS),SPACE=(TRK,(5,5)),UNIT=SYSDA       |
| //CONCAT DD DSN=*.TEMP1,VOL=REF=*.TEMP1,DISP=(OLD,PASS)                 |
| //       DD DSN=*.TEMP2,VOL=REF=*.TEMP2,DISP=(OLD,PASS)                |
| //COMBINE DD SYSOUT=*                                                   |
| //TOOLIN   DD *                                                         |
| * Reformat the File1 records for splicing                               |
|  COPY FROM(IN1) TO(TEMP1) USING(CTL1)                                   |
| * Reformat the File2 records for splicing                               |
|  COPY FROM(IN2) TO(TEMP2) USING(CTL2)                                   |
| * Splice the needed data from File1 and File2 together                  |
|  SPLICE FROM(CONCAT) TO(COMBINE) ON(5,5,ZD) WITH(15,17)                 |
| /*                                                                      |
| //CTL1CNTL DD *                                                         |
|   OUTREC FIELDS=(1,14,      file1 data                                  |
|                 31:X)       add blanks for spliced file2 data           |
| /*                                                                      |
| //CTL2CNTL DD *                                                         |
|   OUTREC FIELDS=(5:1,5,     put file2 key in same place as file1 key    |
|                 15:7,15,    file2 data                                  |
```

```
|                  30:33,2)   file2 data                                  |
| /*                                                                      |
+-------------------------------------------------------------------------+
```

The base records originate from the IN1 data set and are copied and reformatted to the TEMP1 data set. The reformatted TEMP1 records are 31 bytes long and look like this:

```
Y12 89503 MKT
Y12 57301 MKT
Z35 02316 DEV
Y12 91073 MKT
Z35 18693 DEV
```

The overlay records originate from the IN2 data set and are copied and reformatted to the TEMP2 data set. The reformatted TEMP2 records are 31 bytes long and look like this:

```
    89503     27M $9,185,354 CA
    72135     08M   $317,632 MA
    18693     10M $8,732,105 NY
    57301     50M    $30,000 NJ
```

The base and overlay records from the TEMP1 and TEMP2 data sets are sorted and spliced to the COMBINE data set.

The records look like this **after** they are sorted on the 5,5,ZD field, but **before** they are spliced. As a visual aid, the WITH fields in the overlay records are shown in **bold**.

```
Z35 02316 DEV
Z35 18693 DEV
    18693     10M $8,732,105 NY
Y12 57301 MKT
    57301     50M    $30,000 NJ
    72135     08M   $317,632 MA
Y12 89503 MKT
    89503     27M $9,185,354 CA
Y12 91073 MKT
```

The spliced COMBINE records are 31 bytes long and look like this:

```
Z35 18693 DEV 10M $8,732,105 NY
Y12 57301 MKT 50M    $30,000 NJ
Y12 89503 MKT 27M $9,185,354 CA
```

Note that the base records for 18693, 57301 and 89503 have been spliced together with their respective overlay records.

Here's what the various ICETOOL operators do in this job:

The first COPY operator creates reformatted TEMP1 records from the IN1 records. The second COPY operator creates reformatted TEMP2 records from the IN2 records. The TEMP1 records contain the data we want from the IN1 records with blanks where the TEMP2 WITH fields will go. The TEMP2 records have the ON field from IN2 in the same place as in the TEMP1 records, and have the IN2 data where we want it to go in the TEMP1 records. We made the TEMP1 and TEMP2 records the same size so we can use both TEMP1 and TEMP2 as input to the SPLICE operator.

The SPLICE operator sorts the concatenated records from TEMP1 and TEMP2 using the ON field. Concatenation is used to place the TEMP1 records before the TEMP2 records. The spliced records are created from the base

records in TEMP1 and the overlay records in TEMP2. Whenever two records are found with the same ON field, the WITH field from the second record (TEMP2 overlay record) is overlaid on to the first record (TEMP1 base record). The resulting spliced records are written to the COMBINE data set.

Example 2 - Combine complete records from four files

This example shows how you can use the WITHEACH operand to splice complete records from four different data sets together.

```
+-------------------------------------------------------------------+
| //S2    EXEC  PGM=ICETOOL                                         |
| //TOOLMSG   DD  SYSOUT=*                                          |
| //DFSMSG    DD  SYSOUT=*                                          |
| //FILE1 DD DSN=...  input file1 - 300-byte records                |
| //FILE2 DD DSN=...  input file2 - 400-byte records                |
| //FILE3 DD DSN=...  input file3 - 150-byte records                |
| //FILE4 DD DSN=...  input file4 -  20-byte records                |
| //** BE SURE TO USE MOD FOR T1                                    |
| //T1 DD DSN=&TX,UNIT=SYSDA,SPACE=(CYL,(5,5)),                     |
| // DISP=(MOD,PASS)                                                |
| //ALLRCDS DD DSN=...  output file - 870-byte records              |
| //TOOLIN DD *                                                     |
| * Reformat the File1 records for splicing on added sequence number |
|  COPY FROM(FILE1) TO(T1) USING(CTL1)                              |
| * Reformat the File2 records for splicing on added sequence number |
|  COPY FROM(FILE2) TO(T1) USING(CTL2)                              |
| * Reformat the File3 records for splicing on added sequence number |
|  COPY FROM(FILE3) TO(T1) USING(CTL3)                              |
| * Reformat the File4 records for splicing on added sequence number |
|  COPY FROM(FILE4) TO(T1) USING(CTL4)                              |
| * Splice record-by-record on added sequence number                |
|  SPLICE FROM(T1) TO(ALLRCDS) ON(871,8,PD) WITHEACH -              |
|    WITH(301,400) WITH(701,150) WITH(851,20) -                     |
|    USING(CTL5)                                                    |
| /*                                                                |
| //CTL1CNTL DD *                                                   |
| * Reformat records to:                                            |
| * 1          301        701        851        871                |
| * File1bytes|400 blanks|150 blanks|20 blanks |seqno               |
|   OUTREC FIELDS=(1,300,871:SEQNUM,8,PD)                           |
| /*                                                                |
| //CTL2CNTL DD *                                                   |
| * Reformat records to:                                            |
|                                                                   |
| * 1          301        701        851        871                |
| * 300 blanks|File2bytes|150 blanks|20 blanks |seqno               |
|   OUTREC FIELDS=(301:1,400,871:SEQNUM,8,PD)                       |
| /*                                                                |
| //CTL3CNTL DD *                                                   |
| * Reformat records to:                                            |
| * 1          301        701        851        871                |
| * 300 blanks|400 blanks|File3bytes|20 blanks |seqno               |
|   OUTREC FIELDS=(701:1,150,871:SEQNUM,8,PD)                       |
| /*                                                                |
| //CTL4CNTL DD *                                                   |
| * Reformat records to:                                            |
| * 1          301        701        851        871                |
| * 300 blanks|400 blanks|150 blanks|File4bytes|seqno               |
```

```
|   OUTREC FIELDS=(851:1,20,871:SEQNUM,8,PD)                          |
| /*                                                                  |
| //CTL5CNTL DD *                                                     |
| * Remove added sequence number from spliced records to get:         |
| * File1bytes|File2bytes|File3bytes|File4bytes                       |
|  OUTFIL FNAMES=ALLRCDS,OUTREC=(1,870)                               |
| /*                                                                  |
+---------------------------------------------------------------------+
```

Since the data sets do not have a common key, we add sequence numbers to the records from each data set and use the sequence numbers as the ON field for SPLICE. Using this technique, we can splice together the 300-byte records from FILE1, the 400-byte records from FILE2, the 150-byte records from FILE3 and the 20-byte records from FILE4, to produce 870-byte records in ALLRCDS. Conceptually, the 870-byte records in ALLRCDS would look like this:

```
File1 Record1 ... File2Record1 ... File3Record1 ... File4Record1 ...
File1 Record2 ... File2Record2 ... File3Record2 ... File4Record2 ...
...
```

The base records originate from the FILE1 data set and the overlay records originate from the FILE2, FILE3 and FILE4 data sets.

Here's what the various ICETOOL operators do in this job:

The first COPY operator creates reformatted records in the T1 data set with the FILE1 records in positions 1-300, blanks in all other positions up to 870, and a sequence number in positions 871-878. The sequence number will be 1 for the first FILE1 record, 2 for the second FILE1 record, and so on.

The second COPY operator creates reformatted records in the T1 data set with the FILE2 records in positions 301-700, blanks in all other positions up to 870, and a sequence number in positions 871-878. The sequence number will be 1 for the first FILE2 record, 2 for the second FILE2 record, and so on.

The third COPY operator creates reformatted records in the T1 data set with the FILE3 records in positions 701-850, blanks in all other positions up to 870, and a sequence number in positions 871-878. The sequence number will be 1 for the first FILE3 record, 2 for the second FILE3 record, and so on.

The fourth COPY operator creates reformatted records in the T1 data set with the FILE4 records in positions 851-870, blanks in all other positions up to 850, and a sequence number in positions 871-878. The sequence number will be 1 for the first FILE4 record, 2 for the second FILE4 record, and so on.

Note that MOD is used for the T1 data set, so the reformatted records from FILE1, FILE2, FILE3 and FILE4 will be output in that order, ensuring that they are sorted and spliced in that order. Alternatively, you could reformat the FILE1, FILE2, FILE3 and FILE4 data sets to separate data sets which you concatenate together in that order. This technique was used in Example 1.

The SPLICE operator sorts the records from T1 using the sequence number as the ON field. With WITHEACH, the reformatted FILE1 records are treated as the base records, and the reformatted FILE2, FILE3 and FILE4 records are treated as the overlay records; each WITH field is associated with an overlay record in turn. So the first WITH field specifies the bytes to be used from the second duplicate (FILE2 record), the second WITH field specifies the bytes to be used from the third duplicate (FILE3 record) and the third WITH field specifies the bytes to be used from the fourth duplicate (FILE4 record).

SPLICE matches each base and overlay record by their sequence numbers, and creates a new combined 878-byte record. The OUTFIL statement in CTL5CNTL is used to remove the sequence number so that the 870-byte spliced record is written to the ALLRCDS data set.

Example 3 - Create multiple spliced records for each match in two files

This example shows how you can use the WITHALL operand to tell ICETOOL to splice data together for a record from one data set (the first data set) and multiple records from another data set (the second data set) that all have the same ON field (duplicate records). It also shows how to ensure that duplicates from the second data set without a match in the first data set are not written to the output data set.

```
+-------------------------------------------------------------------+
| //S3    EXEC  PGM=ICETOOL                                         |
| //TOOLMSG   DD  SYSOUT=*                                          |
| //DFSMSG    DD  SYSOUT=*                                          |
| //MAST DD *                                                       |
| A0000B0000KRSC0000D000000E0000F00G000                             |
| A1111B1111FLYC1111D111111E1111F11G111                             |
| /*                                                                |
| //UPD DD *                                                        |
| H02KRSI000002J002K002L02                                          |
| H03FLYI000003J003K003L03                                          |
| H04VQXI000004J004K004L04                                          |
| H05FLYI000005J005K005L05                                          |
| H06KHNI000006J006K006L06                                          |
| H07KRSI000007J007K007L07                                          |
| H08FLYI000008J008K008L08                                          |
| H09KHNI000009J009K009L09                                          |
| /*                                                                |
| //TEMP1 DD DSN=&&TEMP1,DISP=(,PASS),SPACE=(TRK,(5,5)),UNIT=SYSDA  |
| //TEMP2  DD DSN=&&TEMP2,DISP=(,PASS),SPACE=(TRK,(5,5)),UNIT=SYSDA |
| //CONCAT DD DSN=*.TEMP1,VOL=REF=*.TEMP1,DISP=(OLD,PASS)           |
| //       DD DSN=*.TEMP2,VOL=REF=*.TEMP2,DISP=(OLD,PASS)          |
| //OUT DD SYSOUT=*                                                 |
| //TOOLIN   DD *                                                   |
| * Reformat the File1 records for splicing.                        |
| * Add 'B' identifier for base records.                            |
|  COPY FROM(MAST) TO(TEMP1) USING(CTL1)                            |
| * Reformat the File2 records for splicing.  Add 'V' identifier.   |
| * Add 'V' identifier for overlay records.                         |
|  COPY FROM(UPD) TO(TEMP2) USING(CTL2)                             |
| * Splice needed base and overlay data together.                   |
| * Do NOT splice identifier.                                       |
|  SPLICE FROM(CONCAT) TO(OUT) WITH(1,7) WITH(13,4) ON(20,3,CH) -   |
|    WITH(23,3) WITH(26,3) WITHALL USING(CTL3)                      |
| /*                                                                |
| //CTL1CNTL DD *                                                   |
| * Set up fields in base records.  Add 'B' id in position 33.      |
|   OUTREC FIELDS=(8:14,5,17:31,3,20:11,3,29:34,4,33:C'B')          |
| /*                                                                |
| //CTL2CNTL DD *                                                   |
| * Set up fields in overlay records.  Add 'V' id in position 33.   |
|   OUTREC FIELDS=(1:7,7,13:18,4,20:4,3,23:1,3,26:22,3,33:C'V')     |
| /*                                                                |
| //CTL3CNTL DD *                                                   |
| * Remove duplicate overlay records without matching base record.  |
| * Remove base or overlay indicator.                               |
|   OUTFIL FNAMES=OUT,OMIT=(33,1,CH,EQ,C'V'),OUTREC=(1,32)          |
| /*                                                                |
+-------------------------------------------------------------------+
```

The base records originate from the MAST data set and are copied and reformatted to the TEMP1 data set. The reformatted TEMP1 records are 33 bytes long and look like this:

```
       C0000    F00KRS      G000B
       C1111    F11FLY      G111B
```

We put a 'B' in position 33 to identify these records as base records.

The overlay records originate from the UPD data set and are copied and reformatted to the TEMP2 data set. The reformatted TEMP2 records are 33 bytes long and look like this:

```
I000002     K002   KRSH02L02    V
I000003     K003   FLYH03L03    V
I000004     K004   VQXH04L04    V
I000005     K005   FLYH05L05    V
I000006     K006   KHNH06L06    V
I000007     K007   KRSH07L07    V
I000008     K008   FLYH08L08    V
I000009     K009   KHNH09L09    V
```

We put a 'V' in position 33 to identify these records as overlay records.

The base and overlay records from the TEMP1 and TEMP2 data sets are sorted and spliced.

The records look like this **after** they are sorted on the 20,3,CH field, but **before** they are spliced. As a visual aid, the WITH fields in the overlay records are shown in **bold**.

```
       C1111    F11FLY      G111B
I000003     K003   FLYH03L03    V
I000005     K005   FLYH05L05    V
I000008     K008   FLYH08L08    V
I000006     K006   KHNH06L06    V
I000009     K009   KHNH09L09    V
       C0000    F00KRS      G000B
I000002     K002   KRSH02L02    V
I000007     K007   KRSH07L07    V
I000004     K004   VQXH04L04    V
```

The spliced output records are 33 bytes long and look like this:

```
I000003C1111K003F11FLYH03L03G111B
I000005C1111K005F11FLYH05L05G111B
I000008C1111K008F11FLYH08L08G111B
I000009     K009   KHNH09L09    V
I000002C0000K002F00KRSH02L02G000B
I000007C0000K007F00KRSH07L07G000B
```

Note that the base record for FLY from the MAST data set has been spliced together with each of the three overlay records for FLY from the UPD data set. Likewise, the base record for KRS from the MAST data set has been spliced together with each of the two overlay records for KRS from the UPD data set.

But also note that the overlay records for KHN have been spliced together. Since KHN does not appear in the MAST data set, we don't want the KHN records to appear in the OUT data set. So we will use the 'V' we put in position 33 for the overlay records to identify and delete spliced overlay records without a matching base record. We only have to do this if we have duplicate overlay records without a matching base record. Single overlay records without a matching base record will be deleted automatically (unless you specify KEEPNODUPS).

After we eliminate the spliced overlay records and the position 33 indicator, the OUT records are 32 bytes long and look like this:

```
I000003C1111K003F11FLYH03L03G111
I000005C1111K005F11FLYH05L05G111
I000008C1111K008F11FLYH08L08G111
I000002C0000K002F00KRSH02L02G000
I000007C0000K007F00KRSH07L07G000
```

Here's what the various ICETOOL operators do in this job:

The first COPY operator creates reformatted TEMP1 records from the MAST records. The second COPY operator creates reformatted TEMP2 records from the UPD records. The TEMP1 records contain the data we want from the MAST records with blanks where the TEMP2 WITH fields will go. The TEMP2 records have the ON field from UPD in the same place as in the TEMP1 records and have the UPD data where we want it to go in the TEMP1 records. We made the TEMP1 and TEMP2 records the same size so we can use both TEMP1 and TEMP2 as input to the SPLICE operator. We added a 'B' at the end of the TEMP1 records to identify them as base records. We added a 'V' at the end of the TEMP2 records to identify them as overlay records.

The SPLICE operator sorts the concatenated records from TEMP1 and TEMP2 using the ON field. The spliced records are created from the base records in TEMP1 and the overlay records in TEMP2. Whenever an overlay record is found with the same ON field as a base record, the overlay record is overlaid on to the base record. The OUTFIL statement in CTL3CNTL is used to remove spliced overlay records, as well as the base/overlay indicator in position 33. The resulting spliced records are written to the OUT data set.

Note that if we had not specified WITHALL, only the first and last records for each set of duplicates would have been spliced, producing the following output:

```
I000008C1111K008F11FLYH08L08G111
I000007C0000K007F00KRSH07L07G000
```

Example 4 - Pull records from a master file in sorted order

This example shows how you can use the WITHALL operand to tell ICETOOL to use ON fields in a "PULL" data set to select one or more records from a "MASTER" data set. In other words, you can use a PULL list to select records from a MASTER list. In this case, the PULL data set has VB records and the MASTER data set has FB records. The ON field is a City Name that can be 1-20 bytes long, and the selected MASTER records are to be sorted by the City Name. (Example 5 shows how to keep the MASTER records in their original order.)

```
+-------------------------------------------------------------------+
| //S4    EXEC  PGM=ICETOOL                                         |
| //TOOLMSG   DD  SYSOUT=*                                          |
| //DFSMSG    DD  SYSOUT=*                                          |
| //PULL DD DSN=VAR.PULL.FILE,DISP=SHR                              |
| //MASTER DD DSN=FIXED.MASTER.FILE,DISP=SHR                        |
| //TEMP1 DD DSN=&T1,DISP=(,PASS),SPACE=(TRK,(5,5)),UNIT=SYSDA      |
| //TEMP2 DD DSN=&T2,DISP=(,PASS),SPACE=(TRK,(5,5)),UNIT=SYSDA      |
| //CONCAT DD DSN=*.TEMP1,VOL=REF=*.TEMP1,DISP=(OLD,PASS)           |
| //       DD DSN=*.TEMP2,VOL=REF=*.TEMP2,DISP=(OLD,PASS)          |
| //OUT DD DSN=FIXED.OUTPUT.FILE,DISP=(NEW,CATLG,DELETE),           |
| //       SPACE=(TRK,(5,5)),UNIT=SYSDA                            |
| //TOOLIN DD *                                                     |
| * Convert PULL records from VB to FB and add 'P' identifier.      |
|  COPY FROM(PULL) USING(CTL1)                                      |
| * Add 'M' identifier to MASTER records.                           |
|  COPY FROM(MASTER) TO(TEMP2) USING(CTL2)                          |
| * Splice PULL and MASTER records (do NOT splice identifier):      |
```

```
| *   Spliced MASTER records with matching PULL records have 'P' id.  |
| *   Spliced MASTER records without matching PULL records            |
|     have 'M' id.                                                    |
| * Eliminate records with 'M' id.                                    |
|  SPLICE FROM(CONCAT) TO(OUT) ON(1,20,CH) WITHALL WITH(1,40) -       |
|    USING(CTL3)                                                      |
| /*                                                                  |
| //CTL1CNTL DD *                                                     |
| * Convert PULL records from VB to FB and add 'P' identifier.        |
|   OUTFIL FNAMES=TEMP1,VTOF,OUTREC=(5,20,41:C'P')                    |
| /*                                                                  |
| //CTL2CNTL DD *                                                     |
| * Add 'M' identifier to MASTER records.                             |
|   OUTREC FIELDS=(1,40,41:C'M')                                      |
| /*                                                                  |
| //CTL3CNTL DD *                                                     |
| * Eliminate MASTER records without matching PULL records.           |
|   OUTFIL FNAMES=OUT,OMIT=(41,1,CH,EQ,C'M'),OUTREC=(1,40)            |
| /*                                                                  |
+---------------------------------------------------------------------+
```

The base records originate from the PULL data set (VAR.PULL.FILE). The PULL data set has variable-length (VB) records with the RDW in positions 1-4 and the variable-length City Name starting in position 5 for 1-20 bytes. Conceptually, the PULL records look like this:

```
Length | Data
    12 | SAN JOSE
    12 | NEW YORK
    11 | DENVER
    15 | LOS ANGELES
```

The overlay records originate from the MASTER data set (FIXED.MASTER.FILE). The MASTER data set has 40-byte fixed-length (FB) records with the City Name in positions 1-20.

The PULL records are copied and reformatted to the TEMP1 data set as 41-byte fixed-length (FB) records with the City Name in positions 1-20 (padded on the right with blanks as necessary), and a 'P' in position 41 to identify them as PULL records. The VTOF and OUTREC options of DFSORT's OUTFIL statement are used to convert the VB records to FB records with blank padding. The reformatted TEMP1 records look like this:

```
SAN JOSE                            P
NEW YORK                            P
DENVER                              P
LOS ANGELES                         P
```

The MASTER records are copied and reformatted to the TEMP2 data set as 41-byte fixed-length (FB) records with an 'M' added in position 41 to identify them as MASTER records. The reformatted TEMP2 records look like this:

```
SAN JOSE            8630  SUSAN       M
PHOENIX             7993  PAUL        M
LOS ANGELES         9203  MICHAEL     M
SAN JOSE            0052  VICKY       M
NEW YORK            5218  CARRIE      M
SAN JOSE            3896  FRANK       M
TUCSON              1056  LISA        M
NEW YORK            6385  MICHAEL     M
PHOENIX             5831  HOLLY       M
```

The base and overlay records from the TEMP1 and TEMP2 data sets are sorted and spliced.

The records look like this **after** they are sorted on the 1,20,CH field, but **before** they are spliced. As a visual aid, the WITH fields in the overlay records are shown in **bold**.

```
DENVER                                P
LOS ANGELES                           P
LOS ANGELES          9203  MICHAEL    M
NEW YORK                              P
NEW YORK             5218  CARRIE     M
NEW YORK             6385  MICHAEL    M
PHOENIX              7993  PAUL       M
PHOENIX              5831  HOLLY      M
SAN JOSE                              P
SAN JOSE             8630  SUSAN      M
SAN JOSE             0052  VICKY      M
SAN JOSE             3896  FRANK      M
TUCSON               1056  LISA       M
```

The spliced records look like this:

```
LOS ANGELES          9203  MICHAEL    P
NEW YORK             5218  CARRIE     P
NEW YORK             6385  MICHAEL    P
PHOENIX              5831  HOLLY      M
SAN JOSE             8630  SUSAN      P
SAN JOSE             0052  VICKY      P
SAN JOSE             3896  FRANK      P
```

Finally, we use the OUTFIL statement for SPLICE to remove each spliced record with an 'M' in position 41, since that represents a base record without a matching overlay record. The OUTFIL statement also removes the 'P' indicator in position 41 from each record since it is not needed in the OUT data set.

Thus, for each MASTER record that matches a PULL record, we've overlaid the PULL record with the MASTER record. This effectively selects all of the MASTER records on the PULL list. The resulting OUT data set (FIXED.OUTPUT.FILE) has the following 40-byte fixed-length records:

```
LOS ANGELES          9203  MICHAEL
NEW YORK             5218  CARRIE
NEW YORK             6385  MICHAEL
SAN JOSE             8630  SUSAN
SAN JOSE             0052  VICKY
SAN JOSE             3896  FRANK
```

Example 5 - Pull records from a master file in their original order

This example is similar to Example 4, except that we want to keep the resulting MASTER records in their original order instead of sorting them by the City Name field. We use DFSORT's SEQNUM option to add a sequence number to each MASTER record before the records are spliced, and we splice that sequence number along with the data. After SPLICE sorts by the City Name, we SORT again by the sequence number to get the resulting MASTER records back in their original order.

```
+-------------------------------------------------------------------+
| //S5    EXEC  PGM=ICETOOL                                         |
| //TOOLMSG   DD  SYSOUT=*                                          |
| //DFSMSG    DD  SYSOUT=*                                          |
| //PULL DD DSN=VAR.PULL.FILE,DISP=SHR                              |
| //MASTER DD DSN=FIXED.MASTER.FILE,DISP=SHR                        |
| //TEMP1 DD DSN=&&TEMP1,DISP=(,PASS),SPACE=(TRK,(5,5)),UNIT=SYSDA  |
| //TEMP2  DD DSN=&&TEMP2,DISP=(,PASS),SPACE=(TRK,(5,5)),UNIT=SYSDA |
```

```
| //CONCAT DD DSN=*.TEMP1,VOL=REF=*.TEMP1,DISP=(OLD,PASS)             |
| //       DD DSN=*.TEMP2,VOL=REF=*.TEMP2,DISP=(OLD,PASS)             |
| //OUT DD DSN=FIXED.OUTPUT.FILE,DISP=(NEW,CATLG,DELETE),             |
| //       SPACE=(TRK,(5,5)),UNIT=SYSDA                               |
| //TOOLIN DD *                                                       |
| * Convert PULL records from VB to FB and add 'P' identifier.        |
|  COPY FROM(PULL) USING(CTL1)                                        |
| * Add sequence number and 'M' identifier to MASTER records.         |
|  COPY FROM(MASTER) TO(TEMP2) USING(CTL2)                            |
| * Splice PULL and MASTER records (splice sequence number, but       |
|   do NOT splice identifier):                                        |
| *   Spliced MASTER records with matching PULL records have 'P' id.  |
| *   Spliced MASTER records without matching PULL records            |
| *   have 'M' id.                                                    |
| * Eliminate records with 'M' id.                                    |
|  SPLICE FROM(CONCAT) TO(TEMP1) ON(1,20,CH) WITHALL WITH(1,48) -     |
|    USING(CTL3)                                                      |
| * Sort resulting spliced records on original sequence number        |
| * to get them back in their original order.                         |
| * Remove id and sequence number.                                    |
|  SORT FROM(TEMP1) TO(OUT) USING(CTL4)                               |
| /*                                                                  |
| //CTL1CNTL DD *                                                     |
| * Convert PULL records from VB to FB and add 'P' identifier         |
|   OUTFIL FNAMES=TEMP1,VTOF,OUTREC=(5,20,49:C'P')                    |
| /*                                                                  |
| //CTL2CNTL DD *                                                     |
| * Add sequence number and 'M' identifier to MASTER records.         |
|   OUTREC FIELDS=(1,40,41:SEQNUM,8,BI,49:C'M')                       |
| /*                                                                  |
| //CTL3CNTL DD *                                                     |
| * Eliminate MASTER records without matching PULL records.           |
|   OUTFIL FNAMES=TEMP1,OMIT=(49,1,CH,EQ,C'M')                        |
| /*                                                                  |
| //CTL4CNTL DD *                                                     |
| * Sort on sequence number and remove id and sequence number.        |
|   SORT FIELDS=(41,8,BI,A)                                           |
|   OUTREC FIELDS=(1,40)                                              |
| /*                                                                  |
+---------------------------------------------------------------------+
```

The resulting OUT data set (FIXED.OUTPUT.FILE) has the following 40-byte fixed-length records:

```
SAN JOSE             8630  SUSAN
LOS ANGELES          9203  MICHAEL
SAN JOSE             0052  VICKY
NEW YORK             5218  CARRIE
SAN JOSE             3896  FRANK
NEW YORK             6385  MICHAEL
```

Example 6 - Create a report showing if needed parts are on-hand

This example shows how you can use the KEEPNODUPS operand to tell ICETOOL to compare the ON fields in a list of needed parts to the ON fields in a list of on-hand parts, and produce a report showing if each needed part is on-hand or not.

```
//S6       EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//ONHAND DD *
P62 Blue
P62 Red
G73 Blue
A27 Green
L90 Red
P63 Blue
/*
//NEEDED DD *
 2003/05/07   A27 Green
 2002/12/29   P62 Blue
 2003/03/17   A27 Blue
 2003/06/14   M92 Yellow
 2002/12/18   L90 Red
/*
//COMBINED DD DSN=&T1,UNIT=SYSDA,SPACE=(TRK,(5,5)),
// DISP=(MOD,PASS)
//RPT  DD SYSOUT=*
//TOOLIN   DD *
* Reformat the ONHAND records for splicing.
* Add 'Yes' for found in ONHAND data set.
* Add 'O' to indicate ONHAND record.
  COPY FROM(ONHAND) TO(COMBINED) USING(CTL1)
* Reformat the NEEDED records for splicing.
* Add 'No' for missing from ONHAND data set.
* Add 'N' to indicate NEEDED record.
  COPY FROM(NEEDED) TO(COMBINED) USING(CTL2)
* Splice ONHAND and NEEDED records (splice identifier):
*   NEEDED records found in ONHAND list will have 'Yes'
*   and 'N'.
*   NEEDED records not found in ONHAND list will have 'No'
*   and 'N'.
*   ONHAND records which are not needed will have 'Yes'
*   and 'O'.
* Eliminate records with 'O'.
  SPLICE FROM(COMBINED) TO(RPT) -
    ON(1,12,CH) WITH(24,10) WITH(40,1) -
    KEEPNODUPS -
    USING(CTL3)
/*
//CTL1CNTL DD *
* Reformat ONHAND records with part in 1-12, 'Yes' in 15-17
* and 'O' in 40.
  OUTREC FIELDS=(1:1,12,15:C'Yes',40:C'O')
/*
//CTL2CNTL DD *
* Reformat NEEDED records with part in 1-12, 'No ' in 15-17,
* date in 24-33 and 'N' in 40.
  OUTREC FIELDS=(1:15,12,15:C'No',24:2,10,40:C'N')
/*
//CTL3CNTL DD *
* Eliminate ONHAND parts that do not appear in NEEDED list.
* Create the report showing if needed parts are on-hand.
  OUTFIL FNAMES=RPT,OMIT=(40,1,CH,EQ,C'O'),OUTREC=(1,33),
```

```
|   HEADER2=(1:'Part',15:'On-Hand',24:'Needed by',/,                  |
|            1:'------------',15:'-------',24:'----------')           |
| /*                                                                  |
+---------------------------------------------------------------------+
```

The base records originate from the ONHAND data set and are copied and reformatted to the COMBINED data set. We put a 'O' in position 40 to identify these records as ONHAND records. The overlay records originate from the NEEDED data set and are copied and reformatted to the COMBINED data set. We put an 'N' in position 40 to identify these records as NEEDED records. Since MODS is used for the COMBINED data set, it contains the reformatted ONHAND records followed by the reformatted NEEDED records. The COMBINED records are 40 bytes long and look like this:

```
P62 Blue      Yes                    O
P62 Red       Yes                    O
G73 Blue      Yes                    O
A27 Green     Yes                    O
L90 Red       Yes                    O
P63 Blue      Yes                    O
A27 Green     No       2003/05/07    N
P62 Blue      No       2002/12/29    N
A27 Blue      No       2003/03/17    N
M92 Yellow    No       2003/06/14    N
L90 Red       No       2002/12/18    N
```

The base and overlay records from the COMBINED data set are sorted and spliced.

The records look like this **after** they are sorted on the 1,12,CH field, but **before** they are spliced. As a visual aid, the WITH fields in the overlay records are shown in **bold**.

```
A27 Blue      No       2003/03/17    N
A27 Green     Yes                    O
A27 Green     No       2003/05/07    N
G73 Blue      Yes                    O
L90 Red       Yes                    O
L90 Red       No       2002/12/18    N
M92 Yellow    No       2003/06/14    N
P62 Blue      Yes                    O
P62 Blue      No       2002/12/29    N
P62 Red       Yes                    O
P63 Blue      Yes                    O
```

The spliced output records are 40 bytes long and look like this:

```
A27 Blue      No       2003/03/17    N
A27 Green     Yes      2003/05/07    N
G73 Blue      Yes                    O
L90 Red       Yes      2002/12/18    N
M92 Yellow    No       2003/06/14    N
P62 Blue      Yes      2002/12/29    N
P62 Red       Yes                    O
P63 Blue      Yes                    O
```

We have three types of records above as follows:

- Records with 'Yes and 'N' are NEEDED records with an ONHAND match that have been spliced together. We want these for our report.

- Records with 'No' and 'N' are NEEDED records without an ONHAND match that have been kept because we used KEEPNODUPS. We want these for our report.
- Records with 'Yes' and 'O' are ONHAND records without a NEEDED match that have been kept because we used KEEPNODUPS. We do not want these for our report.

We use the OUTFIL statement for SPLICE to further process the spliced records. It omits the 'O' records, removes the 'N' byte, and sets up the headers for the report. The resulting RPT data set looks like this:

```
Part          On-Hand  Needed by
------------  -------  ----------
A27 Blue      No       2003/03/17
A27 Green     Yes      2003/05/07
L90 Red       Yes      2002/12/18
M92 Yellow    No       2003/06/14
P62 Blue      Yes      2002/12/29
```

Example 7 - Create a report showing if needed parts are on-hand - advanced

This example is a more complex variation of Example 6. It shows how you can use the WITHALL and KEEPNODUPS operands to tell ICETOOL to compare the ON fields in a list of needed parts to the ON fields in a list of on-hand parts, and produce a report showing if each needed part is on-hand or not. However, it also has duplicate parts in the NEEDED data set, and produces a report with more information from the ONHAND and NEEDED records.

```
+-------------------------------------------------------------------+
| //S7       EXEC PGM=ICETOOL                                       |
| //TOOLMSG  DD SYSOUT=*                                            |
| //DFSMSG   DD SYSOUT=*                                            |
| //ONHAND DD *                                                     |
| P62 Blue           Dallas                                         |
| G73 Blue           San Jose                                       |
| A27 Green          Vancouver                                      |
| /*                                                                |
| //NEEDED DD *                                                     |
|  Rachel       A27 Green       Phoenix                             |
|  Monica       P62 Blue        Phoenix                             |
|  Phoebe       A27 Blue        Toronto                             |
|  Chandler     M92 Yellow      Los Angeles                         |
|  Joey         M92 Yellow      Paris                               |
|  Ross         A27 Green       Paris                               |
| /*                                                                |
| //COMBINED DD DSN=&C1,UNIT=SYSDA,SPACE=(TRK,(5,5)),               |
| // DISP=(MOD,PASS)                                                |
| //TEMP1 DD DSN=&T1,UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(,PASS)      |
| //RPT  DD SYSOUT=*                                                |
| //TOOLIN   DD *                                                   |
| * Reformat the ONHAND records for splicing.                       |
| * Add 'Yes' for found and 'D' for delete record.                  |
|   COPY FROM(ONHAND) TO(COMBINED) USING(CTL1)                      |
| * Extract first record of each set of duplicates from NEEDED.     |
| * Reformat the records for splicing.                              |
| * Add 'No' for missing and 'D' for delete record.                 |
|   SELECT FROM(NEEDED) TO(COMBINED) ON(15,12,CH) -                 |
|     FIRSTDUP USING(CTL2)                                          |
| * Reformat the NEEDED records for splicing.                       |
| * Add 'No' for missing and 'K' for keep record.                   |
|   COPY FROM(NEEDED) TO(COMBINED) USING(CTL3)                      |
```

```
| * Splice ONHAND and NEEDED records.                                  |
| * Splice in Requested by, Ship to and id fields.                     |
| * Eliminate spliced records with 'D'.                                |
|   SPLICE FROM(COMBINED) TO(TEMP1) -                                  |
|     ON(1,12,CH) WITHALL KEEPNODUPS USING(CTL4) -                     |
|     WITH(24,10) WITH(53,13) WITH(66,1)                               |
| * Print report.                                                      |
|   DISPLAY FROM(TEMP1) LIST(RPT) -                                    |
|    INDENT(2) BETWEEN(2) BLANK -                                      |
|    HEADER('Part') ON(1,12,CH) -                                      |
|    HEADER('On-Hand') ON(15,3,CH) -                                   |
|    HEADER('Requested by') ON(24,12,CH) -                             |
|    HEADER('Ship from') ON(38,13,CH) -                                |
|    HEADER('Ship to') ON(53,13,CH)                                    |
| /*                                                                   |
| //CTL1CNTL DD *                                                      |
| * Reformat ONHAND records with Part in 1-12, 'Yes' for found in      |
| * 15-17, From City in 38-50 and 'D' in 66.                           |
|   OUTREC FIELDS=(1:1,12,15:C'Yes',38:20,13,66:C'D')                  |
| /*                                                                   |
| //CTL2CNTL DD *                                                      |
| * Reformat FIRSTDUP records with Part in 1-12, 'No  ' for missing    |
| * in 15-17, Requester Name in 24-35, 'n/a' for From City in 38-40,   |
| * To City in 53-65 and 'D' in 66.                                    |
|   OUTFIL FNAMES=COMBINED,                                            |
|    OUTREC=(1:15,12,15:C'No ',24:2,10,38:C'n/a',                      |
|           53:31,13,66:C'D')                                          |
| /*                                                                   |
| //CTL3CNTL DD *                                                      |
| * Reformat NEEDED records with Part in 1-12, 'No  ' for missing in   |
| * 15-17, Requester Name in 24-35, 'n/a' for From City in 38-40,      |
| * To City in 53-65 and 'K' in 66.                                    |
|   OUTREC FIELDS=(1:15,12,15:C'No ',24:2,10,38:C'n/a',                |
|                 53:31,13,66:C'K')                                    |
| /*                                                                   |
| //CTL4CNTL DD *                                                      |
| * Eliminate ONHAND parts that do not appear in NEEDED list.         |
|   OUTFIL FNAMES=TEMP1,OMIT=(66,1,CH,EQ,C'D')                         |
| /*                                                                   |
+----------------------------------------------------------------------+
```

The base records originate from the ONHAND data set. They are copied and reformatted to the COMBINED data set. The reformatted records look like this:

```
P62 Blue      Yes                   Dallas                    D
G73 Blue      Yes                   San Jose                  D
A27 Green     Yes                   Vancouver                 D
```

We need to make sure that all parts which appear in more than one NEEDED record, but do not appear in the ONHAND list, will appear in the report. For example, we have the following in the NEEDED data set:

```
 Chandler     M92 Yellow      Los Angeles
 Joey         M92 Yellow      Paris
```

But the M92 Yellow part does not appear in the ONHAND data set. If we just let these records be spliced, we will only get one record in the output data set with the M92 Yellow part instead of two. To handle this, we need to set up an extra base record that can be spliced with the two needed records. We get these extra base records for duplicates in NEEDED with the SELECT operator, by specifying the FIRSTDUP operand. The FIRSTDUP records

are sorted on the 1,12,CH field and reformatted to the COMBINED data set. The reformatted records look like this:

```
A27 Green     No       Rachel        n/a           Phoenix      D
M92 Yellow    No       Chandler      n/a           Los Angeles  D
```

The overlay records originate from the NEEDED data set and are copied and reformatted to the COMBINED data set. The reformatted records look like this:

```
A27 Green     No       Rachel        n/a           Phoenix      K
P62 Blue      No       Monica        n/a           Phoenix      K
A27 Blue      No       Phoebe        n/a           Toronto      K
M92 Yellow    No       Chandler      n/a           Los Angeles  K
M92 Yellow    No       Joey          n/a           Paris        K
A27 Green     No       Ross          n/a           Paris        K
```

The base and overlay records from the COMBINED data set are sorted and spliced.

The records look like this **after** they are sorted on the 1,12,CH field, but **before** they are spliced. As a visual aid, the WITH fields in the overlay records are shown in **bold**.

```
A27 Blue      No       Phoebe        n/a           Toronto      K
A27 Green     Yes                    Vancouver                  D
A27 Green     No       Rachel        n/a           Phoenix      D
A27 Green     No       Rachel        n/a           Phoenix      K
A27 Green     No       Ross          n/a           Paris        K
G73 Blue      Yes                    San Jose                   D
M92 Yellow    No       Chandler      n/a           Los Angeles  D
M92 Yellow    No       Chandler      n/a           Los Angeles  K
M92 Yellow    No       Joey          n/a           Paris        K
P62 Blue      Yes                    Dallas                     D
P62 Blue      No       Monica        n/a           Phoenix      K
```

The spliced records look like this:

```
A27 Blue      No       Phoebe        n/a           Toronto      K
A27 Green     Yes      Rachel        Vancouver     Phoenix      D
A27 Green     Yes      Rachel        Vancouver     Phoenix      K
A27 Green     Yes      Ross          Vancouver     Paris        K
G73 Blue      Yes                    San Jose                   D
M92 Yellow    No       Chandler      n/a           Los Angeles  K
M92 Yellow    No       Joey          n/a           Paris        K
P62 Blue      Yes      Monica        Dallas        Phoenix      K
```

Records with 'D' are not needed, so we use the OUTFIL statement for SPLICE to omit them. The TEMP1 records look like this:

```
A27 Blue      No       Phoebe        n/a           Toronto      K
A27 Green     Yes      Rachel        Vancouver     Phoenix      K
A27 Green     Yes      Ross          Vancouver     Paris        K
M92 Yellow    No       Chandler      n/a           Los Angeles  K
M92 Yellow    No       Joey          n/a           Paris        K
P62 Blue      Yes      Monica        Dallas        Phoenix      K
```

Although we could have used the OUTFIL statement for SPLICE to print the report, we've chosen instead to use a separate DISPLAY operator. DISPLAY requires an extra pass over the spliced records in TEMP1, but is easier to use than OUTFIL for reports. The resulting RPT data set looks like this:

```
Part         On-Hand  Requested by  Ship from      Ship to
-----------  -------  ------------  -------------  -------------
A27 Blue     No       Phoebe        n/a            Toronto
A27 Green    Yes      Rachel        Vancouver      Phoenix
A27 Green    Yes      Ross          Vancouver      Paris
M92 Yellow   No       Chandler      n/a            Los Angeles
M92 Yellow   No       Joey          n/a            Paris
P62 Blue     Yes      Monica        Dallas         Phoenix
```

OUTFIL Enhancements

This section discusses the following new features of DFSORT's powerful OUTFIL multiple output and reporting control statement:

- SAMPLE=n and SAMPLE=(n,m) options
- REPEAT=n option
- SPLITBY=n option

"Reformatting Enhancements" on page 29 discusses additional enhancments to OUTFIL's OUTREC option.

"INCLUDE and OMIT Enhancements" on page 34 discusses additional enhancments to OUTFIL's INCLUDE and OMIT options.

Refer to Chapter 3 of *DFSORT Application Programming Guide* for general information on DFSORT's OUTFIL statement, and in particular for the framework of existing OUTFIL features upon which these new features are built.

SAMPLE

New SAMPLE=n and SAMPLE=(n,m) options of OUTFIL allow you to sample records in a variety of ways. SAMPLE=n and SAMPLE=(n,m) are optional, and mutually exclusive, for OUTFIL.

General Description

OUTFIL's STARTREC option can be used to start processing for an OUTFIL group at a specific OUTFIL input record. OUTFIL's ENDREC option can be used to end processing for an OUTFIL group at a specific OUTFIL input record. OUTFIL's new SAMPLE option can be used to select a sample of OUTFIL input records for an OUTFIL group using a specific interval and number of records in that interval. Separately or together, STARTREC, ENDREC, and SAMPLE can be used to select a range of records to which subsequent OUTFIL processing will apply.

Option Description

SAMPLE=n and SAMPLE=(n,m)

Specifies a sample of OUTFIL input records to be processed for this OUTFIL group. The sample consists of the first m records in every nth interval.

n specifies the interval size. The value for n starts at 2 (sample every other record) and is limited to 28 digits (15 significant digits).

m specifies the number of records to be processed in each interval. The value for m starts at 1 (process the first record in each interval) and is limited to 28 digits (15 significant digits). If m is not specified, 1 is used for m. If m is specified, it must be less than n.

OUTFIL Example with SAMPLE

```
+-----------------------------------------------------------+
| * Process records 1, 6, 11, ...                           |
|   OUTFIL FNAMES=OUT1,SAMPLE=5                             |
|                                                           |
| * Process records 1, 2, 1001, 1002, 2001, 2002            |
|   OUTFIL FNAMES=OUT2,SAMPLE=(1000,2),ENDREC=2500          |
|                                                           |
| * Process records 23, 48, 73                              |
|   OUTFIL FNAMES=OUT3,STARTREC=23,ENDREC=75,SAMPLE=25      |
|                                                           |
| * Process records 1001, 1002, 1003, 1101, 1102, 1103, ... |
|   OUTFIL FNAMES=OUT4,STARTREC=1001,SAMPLE=(100,3)         |
+-----------------------------------------------------------+
```

REPEAT

A new REPEAT=n option of OUTFIL allows you to write each output record multiple times. REPEAT=n is optional for OUTFIL.

General Description

OUTFIL's new REPEAT option can be used to repeat each OUTFIL output record a specified number of times. OUTFIL OUTREC's SEQNUM option can be used to assign a different sequence number to each repeated record.

Option Description

REPEAT=n

Specifies the number of times each OUTFIL output record is to be repeated for this OUTFIL group. Each OUTFIL output record is written n times.

If SEQNUM is used in the OUTREC parameter for this OUTFIL group, the sequence number will be incremented for each repeated record. For example, if your input is:

```
RECORD A
RECORD B
```

and you specify:

```
OUTFIL OUTREC=(1,8,X,SEQNUM,5,ZD),REPEAT=2
```

your output will be:

```
RECORD A 00001
RECORD A 00002
RECORD B 00003
RECORD B 00004
```

If you specify REPEAT=n with / in OUTFIL OUTREC, the first line is written n times, then the second line is written n times, and so on. If SEQNUM is used, all lines for the same record are given the same sequence number. For example, if your input is:

```
RECORD A
RECORD B
```

and you specify:

```
OUTFIL OUTREC=(C'P1>',X,1,6,X,SEQNUM,4,ZD,/,
  C'P2>',X,8,1,X,SEQNUM,4,ZD),REPEAT=2
```

your output will be:

```
P1> RECORD 0001
P1> RECORD 0002
P2> A 0001
P2> A 0002
P1> RECORD 0003
P1> RECORD 0004
P2> B 0003
P2> B 0004
```

The REPEAT parameter cannot be used with any of the report parameters (LINES, HEADER1, TRAILER1, HEADER2, TRAILER2, SECTIONS, and NODETAIL).

n specifies the number of times each OUTFIL output record is to be repeated. The value for n starts at 2 (write record twice) and is limited to 28 digits (15 significant digits).

OUTFIL Example with REPEAT

```
+--------------------------------------------------------------------+
| * Write each output record 12 times.                               |
|   OUTFIL FNAMES=OUT1,REPEAT=12                                     |
|                                                                    |
| * Write each included and reformatted output record 50 times.      |
| * (The sequence number will be incremented for each repetition.)   |
|   OUTFIL FNAMES=OUT2,INCLUDE=(5,2,SS,EQ,C'B2,C5,M3'),              |
|     OUTREC=(1,20,40X'FF',SEQNUM,5,ZD),REPEAT=50                    |
+--------------------------------------------------------------------+
```

SPLITBY

A new SPLITBY=n option of OUTFIL allows you to write groups of records in rotation among multiple output data sets. SPLIT and SPLITBY=n are optional, and mutually exclusive, for OUTFIL.

General Description

OUTFIL's SPLIT option can be used to distribute one record at a time among the OUTFIL data sets. OUTFIL's new SPLITBY option can be used to distribute multiple records at a time among the OUTFIL data sets.

With SPLIT, the first output record is written to the first OUTFIL data set in the group, the second output record is written to the second data set, and so on. When each OUTFIL data set has one record, the rotation starts again with the first OUTFIL data set.

SPLITBY can be used to rotate by a specified number of records rather than by one record, for example, records 1-10 to the first OUTFIL data set, records 11-20 to the second OUTFIL data set, and so on.

Option Descriptions

SPLIT

Splits the output records one record at a time in rotation among the data sets of this OUTFIL group until all of the output records have been written. As a result, the records will be split as evenly as possible among all of the data sets in the group.

As an example, for an OUTFIL group with three data sets:

- the first OUTFIL data set in the group will receive records 1, 4, 7, and so on.
- the second OUTFIL data set in the group will receive records 2, 5, 8, and so on.

- the third OUTFIL data set in the group will receive records 3, 6, 9, and so on.

SPLIT is equivalent to SPLITBY=1.

The SPLIT parameter cannot be used with any of the report parameters (LINES, HEADER1, TRAILER1, HEADER2, TRAILER2, SECTIONS, and NODETAIL).

SPLITBY=n
Splits the output records n records at a time in rotation among the data sets of this OUTFIL group until all of the output records have been written.

As an example, if SPLITBY=10 is specified for an OUTFIL group with three data sets:

- the first OUTFIL data set in the group will receive records 1-10, 31-40, and so on.
- the second OUTFIL data set in the group will receive records 11-20, 41-50, and so on.
- the third OUTFIL data set in the group will receive records 21-30, 51-60, and so on.

SPLITBY=1 is equivalent to SPLIT.

The SPLITBY parameter cannot be used with any of the report parameters (LINES, HEADER1, TRAILER1, HEADER2, TRAILER2, SECTIONS, and NODETAIL).

n specifies the number of records to split by. The value for n starts at 1 and is limited to 28 digits (15 significant digits).

OUTFIL Example with SPLIT and SPLITBY

```
+--------------------------------------------------------------------+
| * Write record 1 to PIPE1, record 2 to PIPE2, record 3 to PIPE3,   |
| * record 4 to PIPE4, record 5 to PIPE1, record 6 to PIPE2,         |
| * and so on.                                                       |
|  OUTFIL FNAMES=(PIPE1,PIPE2,PIPE3,PIPE4),SPLIT                    |
|                                                                    |
| * Split the included and reformatted output records evenly         |
| * between TAPE1 and TAPE2.                                        |
|  OUTFIL FNAMES=(TAPE1,TAPE2),SPLIT,                                |
|    INCLUDE=(8,2,ZD,EQ,27),OUTREC=(5X,1,75)                         |
|                                                                    |
| * Write records 1-5 to PIPEA, records 6-10 to PIPEB,               |
| * records 11-15 to PIPEC, records 16-20 to PIPED,                  |
| * records 21-25 to PIPEA, records 26-30 to PIPEB, and so on.       |
|  OUTFIL FNAMES=(PIPEA,PIPEB,PIPEC,PIPED),SPLITBY=5                 |
|                                                                    |
| * Split the included and reformatted output records               |
| * 100 at a time between TAPE3 and TAPE4.                          |
|  OUTFIL FNAMES=(TAPE3,TAPE4),SPLITBY=100,                          |
|    INCLUDE=(8,2,ZD,EQ,27),OUTREC=(5X,1,75)                         |
+--------------------------------------------------------------------+
```

Reformatting Enhancements

This section discusses the following new features of DFSORT's INREC, OUTREC and OUTFIL OUTREC reformatting control statements:

- Arithmetic expressions and decimal constants
- DATE4 constant

Refer to Chapter 3 of *DFSORT Application Programming Guide* for general information on DFSORT's INREC, OUTREC and OUTFIL statements, and in particular for the framework of existing reformatting features upon which these new features are built.

Arithmetic Expressions and Decimal Constants

INREC, OUTREC and OUTFIL OUTREC now allow you to insert decimal constants and arithmetic expressions in your records as numeric or edited character values.

General Description

You can now optionally choose to include the following in your reformatted INREC, OUTREC and OUTFIL OUTREC records:

- Decimal constants (+n and -n) converted to BI, FI, PD, ZD or FS numeric values, or to CH values edited to contain signs, thousands separators, decimal points, leading zeros or no leading zeros, and so on.
- The results of arithmetic expressions combining fields (p,m,f), decimal constants (+n and -n), operators (MIN, MAX, MUL, DIV, MOD, ADD and SUB) and parentheses, converted to BI, FI, PD, ZD or FS numeric values, or to CH values edited to contain signs, thousands separators, decimal points, leading zeros or no leading zeros, and so on.

Option Descriptions

p,m,f,edit or (p,m,f),edit
(p,m,f),edit can now be used as the equivalent of p,m,f,edit. See the description of "p,m,f,edit" under OUTFIL OUTREC in Chapter 3 of *DFSORT Application Programming Guide* for details on p,m,f,edit.

p,m,f,to or (p,m,f),to
(p,m,f),to can now be used as the equivalent of p,m,f,to. See the description of "p,m,f,to" under OUTFIL OUTREC in Chapter 3 of *DFSORT Application Programming Guide* for details on p,m,f,to.

deccon,edit or (deccon),edit
Specifies that an edited decimal constant is to appear in the reformatted output record. The decimal constant must be in the form +n or -n where n is 1 to 15 decimal digits. **The sign (+ or -) must be specified**. A decimal constant produces a signed, 15-digit zoned decimal (ZD) result to be edited using the available edit options (M0-M26, EDIT, EDxy, SIGNS, SIGNz, LENGTH) as specified. If an Mn, EDIT, or EDxy parameter is not specified, the decimal constant is edited using the M0 edit mask.

See "p,m,f,edit" under OUTFIL OUTREC in Chapter 3 of *DFSORT Application Programming Guide* for further details on the edit options you can use.

Sample Syntax:

```
OUTREC FIELDS=(5,8,+4096,2X,-17,M18,LENGTH=7,2X,
  (+2000000),EDIT=(STTTTT.TT),SIGNS=(+))
```

deccon,to or (deccon),to
Specifies that a converted decimal constant is to appear in the reformatted output record. The decimal constant must be in the form +n or -n where n is 1 to 15 decimal digits. **The sign (+ or -) must be specified** A decimal constant produces a signed, 15-digit zoned decimal (ZD) result to be converted to BI, FI, ZD, PD or FS format using the available to options (TO, LENGTH) as specified.

See "p,m,f,to" under OUTFIL OUTREC in Chapter 3 of *DFSORT Application Programming Guide* for further details on the to options you can use.

Sample Syntax:

```
OUTFIL FNAMES=OUT1,
 OUTREC=(6:+0,TO=PD,LENGTH=6,+0,TO=PD,LENGTH=6,/,
        6:(-4096),TO=ZD,LENGTH=12)
```

arexp,edit or (arexp),edit

Specifies that the edited result of an arithmetic expression is to appear in the reformatted output record. An arithmetic expression takes the form:

```
term,operator,term<,operator,...>
```

where:

- **term** is a field (p,m,f) or a decimal constant (+n or -n). You can use BI, FI, PD, PD0, ZD, FS, DTn and TMn fields in arithmetic expressions as described for "p,m,f,edit" under OUTFIL OUTREC in Chapter 3 of *DFSORT Application Programming Guide*. See "deccon,edit" above for details on the decimal constants you can use.
- **operator** is MIN (minimum), MAX (maximum), MUL (multiplication), DIV (division), MOD (modulus), ADD (addition) or SUB (subtraction).

  The order of evaluation precedence for the operators is as follows unless it is changed by parentheses:

  1. MIN and MAX
  2. MUL, DIV and MOD
  3. ADD and SUB

The intermediate or final result of a DIV operation is rounded down to the nearest integer. The intermediate or final result of a MOD operation is an integer remainder with the same sign as the dividend. If an intermediate or final result of an arithmetic expression overflows 15 digits, the overflowing intermediate or final result will be truncated to 15 digits, intentionally or unintentionally. If an intermediate or final result of an arithmetic expression requires division or modulus by 0, the intermediate or final result will be set to 0, intentionally or unintentionally.

An arithmetic expression produces a signed, 15-digit zoned decimal (ZD) result to be edited using the available edit options (M0-M26, EDIT, EDxy, SIGNS, SIGNz, LENGTH) as specified. If an Mn, EDIT, or EDxy parameter is not specified, the result is edited using the M0 edit mask.

See "p,m,f,edit" under OUTFIL OUTREC in Chapter 3 of *DFSORT Application Programming Guide* for further details on the edit options you can use.

Sample Syntax:

```
INREC FIELDS=(5:C'% REDUCTION FOR ',21,8,C' IS ',
  ((11,6,ZD,SUB,31,6,ZD),MUL,+1000),DIV,11,6,ZD,
  EDIT=(SIIT.T),SIGNS=(+,-))
```

arexp,to or (arexp),to

Specifies that the converted result of an arithmetic expression is to appear in the reformatted output record. See "arexp,edit" above for further details on arithmetic expressions.

An arithmetic expression produces a signed, 15-digit zoned decimal (ZD) result to be converted to BI, FI, ZD, PD or FS format using the available to options (TO, LENGTH) as specified.

See "p,m,f,to" under OUTFIL OUTREC in Chapter 3 of *DFSORT Application Programming Guide* for further details on the to options you can use.

Sample Syntax:

```
OUTFIL FNAMES=OUT,
  OUTREC=(61,3,X,
          35,6,FS,ADD,45,6,FS,ADD,55,6,FS,TO=FS,LENGTH=7,X,
          (5,3,PD,MIN,11,3,PD),TO=PD,LENGTH=3,X,
          64,5,SEQNUM,5,ZD)
```

Symbols

The following operator keywords are not allowed as symbols: ADD, DIV, MAX, MIN, MOD, MUL and SUB.

You can use symbols for fields and decimal constants in arithmetic expressions. **The sign (+ or -) must be specified** for decimal constants.

For example:

```
...
//SYMNAMES DD *
Part1_Count,1,4,BI
Part2_Count,*,4,BI
Master_Count,28,6,ZD
Used_Count,*,5,ZD
Plus1000,+1000
Stopper,+999
/*
//SYSIN DD *
 OUTREC FIELDS=(21:Part1_Count,MIN,Part2_Count,M25,LENGTH=8,
   35:(Master_Count,SUB,Used_Count),DIV,Plus1000,TO=ZD,
   Stopper,TO=PD,LENGTH=2)
...
/*
```

The OUTREC statement will be transformed to:

```
 OUTREC FIELDS=(21:1,4,BI,MIN,5,4,BI,M25,LENGTH=8,35:(28,6,ZD,SUB,34,5,*
               ZD),DIV,+1000,TO=ZD,+999,TO=PD,LENGTH=2)
```

OUTREC Example with Expressions

```
+-----------------------------------------------------+
|   OUTREC FIELDS=(1,20,                              |
|     (5,4,FI,ADD,3,2,FI,ADD,23,2,FI),DIV,+1000,      |
|        EDIT=(STTTTTTT),SIGNS=(,-),2X,               |
|     9,5,ZD,MIN,16,5,FS,TO=ZD,LENGTH=5,2X,           |
|     21,40,                                          |
|     42,2,PD,MUL,-1,TO=PD,LENGTH=2)                  |
+-----------------------------------------------------+
```

This example illustrates how input records can be reformatted for output to contain the results of arithmetic expressions involving input fields, decimal constants, operators and parentheses.

The reformatted output records look as follows:

Position	Contents
1-20	Input positions 1 through 20
21-28	A CH field containing the total of the FI fields from positions 5 through 8, 3 through 4 and 23 through 24, divided by 1000, and edited according to the specified edit pattern.
29-30	EBCDIC blanks
31-35	A ZD field containing the minimum of the ZD field in positions 9 through 13 and the FS field in positions 16 through 20.
36-37	EBCDIC blanks
38-77	Input positions 21-60
78-79	The PD field in positions 42 through 43 with the sign reversed (for example, +123 converted to -123 or -987 converted to +987).

DATE4 Constant

A new DATE4 option for INREC, OUTREC and OUTFIL OUTREC allows you to insert a timestamp for your DFSORT run in your records in the form 'yyyy-mm-dd-hh.mm.ss'.

General Description

You can now optionally choose to include a CH timestamp representing the date of the run as a separation field in your INREC, OUTREC and OUTFIL OUTREC records.

The DATE4 keyword is not allowed as a symbol.

Option Description

DATE4

Constant for current date and time. The date and time of the run is to appear in the reformatted output records. The Table below shows the constant generated for DATE4, its length and an example.

Separation Field	Constant	Length (bytes)	April 19, 2003, 04:52:45 PM
DATE4	C'yyyy-mm-dd-hh.mm.ss'	19	C'2003-04-19-16.52.45'

yyyy represents the year, mm (for date) represents the month (01-12), dd represents the day (01-31), hh represents the hour (00-23), mm (for time) represents the minutes (00-59), and ss represents the seconds (00-59).

OUTREC Example with DATE4

```
OUTREC FIELDS=(C'Creation date:',X,DATE4,X,1,200)
```

This example illustrates how input records can be reformatted for output to contain a DATE4 timestamp.

The reformatted output records look as follows:

Position	Contents
1-14	The character string 'Creation date:'.
15	An EBCDIC blank.
16-34	The character string C'yyyy-mm-dd-hh.mm.ss' where yyyy-mm-dd is the date of the run, and hh.mm.ss is the time of the run.
35	An EBCDIC blank.
36-235	Input positions 1-200.

INCLUDE and OMIT Enhancements

This section discusses the following new features of DFSORT's INCLUDE, OMIT, OUTFIL INCLUDE and OUTFIL OMIT filtering control statements:

- Substring length up to 32752
- DATE4 constant
- PD0 format

Refer to Chapter 3 of *DFSORT Application Programming Guide* for general information on DFSORT's INCLUDE, OMIT and OUTFIL statements, and in particular for the framework of existing filtering features upon which these new features are built.

Substring Length up to 32752

The maximum length for an SS (substring) field used with INCLUDE, OMIT, OUTFIL INCLUDE and OUTFIL OMIT has been raised from 256 bytes to 32752 bytes.

General Description

The substring comparison test available with the INCLUDE, OMIT, OUTFIL INCLUDE and OUTFIL OMIT statements now allows you to include or omit records which have a specified character or hexadecimal constant anywhere within a field that can be up to 32752 bytes long. Thus, you can check for a constant anywhere in a large field, or even anywhere in an entire record, more easily. For example, you can use an SS condition in an INCLUDE statement to keep only those 25000-byte records that have C'OK' somewhere in the record.

INCLUDE Example with SS

```
+-----------------------------------------------+
|   INCLUDE FORMAT=SS,                          |
|     COND=(11,10000,EQ,C'Error',OR,            |
|           11,10000,EQ,C'Warning')             |
+-----------------------------------------------+
```

This example illustrates how to include only records in which the character string 'Error' or 'Warning' is found somewhere within bytes 11 to 10010.

DATE4 Constant

A new DATE4 option of INCLUDE, OMIT, OUTFIL INCLUDE and OUTFIL OMIT allows you to compare a field to a timestamp for your DFSORT run in the form 'yyyy-mm-dd-hh.mm.ss', or to a portion of that timestamp truncated on the right.

General Description

The field-to-constant comparison test available with the INCLUDE, OMIT, OUTFIL INCLUDE and OUTFIL OMIT statements now allows you to use DATE4 as a constant representing the date and time of a run. DATE4 can be compared to a BI, CH, AC, AQ or D2 field.

The DATE4 keyword is not allowed as a symbol.

Option Description

DATE4

Generates a character string for the date and time of the run. The Table below shows the character string generated for DATE4, and an example.

```
+-----------+-----------------------+-----------------------------+
| Operand   | Constant              | April 19, 2003, 04:52:45 PM |
+-----------+-----------------------+-----------------------------+
| DATE4     | C'yyyy-mm-dd-hh.mm.ss' | C'2003-04-19-16.52.45'      |
+-----------+-----------------------+-----------------------------+
```

yyyy represents the year, mm (for date) represents the month (01-12), dd represents the day (01-31), hh represents the hour (00-23), mm (for time) represents the minutes (00-59), and ss represents the seconds (00-59).

INCLUDE Example with DATE4

```
+-------------------------------------------------+
|   INCLUDE COND=(1,19,CH,GT,DATE4)               |
+-------------------------------------------------+
```

This example illustrates how to include only those records with a C'yyyy-mm-dd-hh.mm.ss' date value in positions 1-19 greater than the DATE4 character string for the run.

Note: When a field is shorter than the DATE4 character string it's compared to, DFSORT truncates the DATE4 string on the right. You can take advantage of this to compare a field to only part of the DATE4 timestamp when appropriate. For example:

```
INCLUDE COND=(1,13,CH,GT,DATE4)
```

would compare the field in positions 1-13 to the truncated DATE4 constant C'yyyy-mm-dd-hh'.

PD0 Format

A PD0 field can now be compared to a hexadecimal constant or to another PD0 field for INCLUDE, OMIT, OUTFIL INCLUDE and OUTFIL OMIT.

General Description

The field-to-field and field-to-constant comparison tests available with the INCLUDE, OMIT, OUTFIL INCLUDE and OUTFIL OMIT statements now allow you to use PD0 fields. A 2-8 byte PD0 field can be compared to another 2-8 byte PD0 field, or to a hexadecimal constant. Refer to "Appendix C: Data Format Descriptions" in *DFSORT Application Programming Guide* for information on PD0 format fields.

Option Description

p,m,PD0

PD0 compare field. p is the position and m is the length. The Table below shows the acceptable lengths and a description for PD0 compare fields.

Format Code	Length	Description
PD0	2 to 8 bytes	Packed decimal with sign and first digit ignored

Since the first digit and sign are ignored in a PD0 field, you should not include the first digit or sign in a hexadecimal constant to be compared to a PD0 field. For example, 3-byte PD0 values like X'01234C' and X'01234D' would be equal to a hexadecimal constant of X'1234'.

In a PD0 field-to-field comparison with an unequal number of decimal digits in the fields, the shorter PD0 field is padded on the right with decimal zeros.

In a PD0 field-to-constant comparison with an unequal number of decimal digits, the hexadecimal constant is truncated, or padded on the right with decimal zeros, to the number of digits in the PD0 field.

INCLUDE Example with PD0

```
INCLUDE FORMAT=PD0,
   COND=(15,5,GE,X'34567890',OR,
        15,5,EQ,28,5)
```

This example illustrates how to include only those records with a PD0 value in positions 15-19 greater than 34567890, or with a PD0 value in positions 15-19 equal to a PD0 value in positions 28-32.

FORMAT Enhancement

FORMAT=f can now be used with mixed p,m and p,m,f fields for SORT, MERGE, SUM, INCLUDE and OMIT. f from FORMAT=f will be used for p,m fields but not for p,m,f fields.

General Description

FORMAT=f can now be used to specify a particular format for compare fields (INCLUDE or OMIT statement), control fields (SORT or MERGE statement), and sum fields (SUM statement) specified without a format (p,m) even when other such fields are specified with a format (p,m,f). (p is position, m is length and f is format.)

f from FORMAT=f is used for p,m fields. f from FORMAT=f is ignored for p,m,f fields.

For example:

- The following INCLUDE statements are all equivalent.

```
* p,m,f used for all fields - FORMAT=f not specified.
  INCLUDE COND=(5,5,ZD,EQ,12,3,PD,OR,21,3,PD,NE,35,5,ZD)

* FORMAT=ZD used for p,m fields, but not for p,m,PD fields.
  INCLUDE FORMAT=ZD,COND=(5,5,EQ,12,3,PD,OR,21,3,PD,NE,35,5)

* FORMAT=PD used for p,m fields, but not for p,m,ZD fields.
  INCLUDE COND=(5,5,ZD,EQ,12,3,OR,21,3,NE,35,5,ZD),FORMAT=PD
```

- The following SORT statements are all equivalent.

```
* p,m,f used for all fields - FORMAT=f not specified.
  SORT FIELDS=(5,5,ZD,A,12,6,PD,D,21,3,PD,A,35,7,ZD,A)

* FORMAT=ZD used for p,m fields, but not for p,m,PD fields.
  SORT FORMAT=ZD,FIELDS=(5,5,A,12,6,PD,D,21,3,PD,A,35,7,A)

* FORMAT=PD used for p,m fields, but not for p,m,ZD fields.
  SORT FIELDS=(5,5,ZD,A,12,6,D,21,3,A,35,7,ZD,A),FORMAT=PD
```

- The following SUM statements are all equivalent.

```
* p,m,f used for all fields - FORMAT=f not specified.
  SUM FIELDS=(5,5,ZD,12,6,PD,21,3,PD,35,7,ZD)

* FORMAT=ZD used for p,m fields, but not for p,m,PD fields.
  SUM FORMAT=ZD,FIELDS=(5,5,12,6,PD,21,3,PD,35,7)

* FORMAT=PD used for p,m fields, but not for p,m,ZD fields.
  SUM FIELDS=(5,5,ZD,12,6,21,3,35,7,ZD),FORMAT=PD
```

For an INCLUDE, OMIT, SORT, MERGE or SUM statement, FORMAT=f must be specified if any field is specified as p,m rather than p,m,f. DFSORT issues an informational message and ignores FORMAT=f if all of the fields are specified as p,m,f.

INCLUDE Example with p,m and p,m,f fields

```
+-----------------------------------------------------------+
|  INCLUDE  FORMAT=CH,                                      |
|    COND=((5,1,EQ,8,1),&,                                  |
|          ((20,1,EQ,C'A',&,30,1,FI,GT,10),|,               |
|           (20,1,EQ,C'B',&,30,1,FI,LT,100),|,              |
|           (20,1,NE,C'A',&,20,1,NE,C'B')))                 |
+-----------------------------------------------------------+
```

This example illustrates how to only include records in which byte 5 equals byte 8 AND at least one of the following is true:

- Byte 20 equals 'A' and byte 30 is greater than 10
- Byte 20 equals 'B' and byte 30 is less than 100
- Byte 20 is not equal to 'A' or 'B'.

Note that p,m,FI is used for the FI fields, and p,m with FORMAT=CH is used for all of the CH fields. With FORMAT=f, you can mix p,m and p,m,f fields when that's convenient such as when all or most of the fields have the same format (although you can always code p,m,f for all fields and not use FORMAT=f, if you prefer).

Continuation Enhancement

Enhancements to DFSORT's control statement continuation rules allow you to continue a line that breaks at column 71 anywhere in columns 2 to 16 of the next line.

General Description

A continuation line is treated as a logical extension of the preceding line. Either an operand or a remark field can begin on one line (referred to as "line 1" in the bullets below) and continue on the next line (referred to as "line 2" in the bullets below). The following are the rules for continuation illustrated with examples (these different types of continuation can be intermixed):

Implicit continuation in 2-71: If line 1 breaks at a comma-blank or semicolon-blank or colon-blank, DFSORT continues on line 2 with the first nonblank character it finds in columns 2-71. For example:

```
*        1         2         3         4         5         6         7
*234567890123456789012345678901234567890123456789012345678901234567890123456789012
 INCLUDE COND=(5,4,CH,EQ,
  C'ABCD')
 SORT FIELDS=(9,
              3,
              ZD,
              A)
 OUTREC FIELDS=(1,27,2X,       FIRST FIELD AND TWO BLANKS
                51,2,BI,M11,   SECOND FIELD
                60:9,3,ZD,PD)  THIRD FIELD
```

The above statements will be treated as if they were specified as:

```
 INCLUDE COND=(5,4,CH,EQ,C'ABCD')
 SORT FIELDS=(9,3,ZD,A)
 OUTREC FIELDS=(1,27,2X,51,2,BI,M11,60:9,3,ZD,PD)
```

Explicit continuation in 16: If line 1 breaks at column 71 with a nonblank in column 72, and columns 2-15 of line 2 are blank, DFSORT continues on line 2 with whatever character it finds in column 16 (blank or nonblank). For example:

```
*        1         2         3         4         5         6         7
*234567890123456789012345678901234567890123456789012345678901234567890123456789012
                                                 INCLUDE COND=(5,4,CH,E*
               Q,C'ABCD')
                                                      SORT FIELDS=(9,3,*
               ZD,A)
                                OUTREC FIELDS=(1,80,C'BLANK WITHIN A*
                LITERAL')
```

The above statements will be treated as if they were specified as:

```
 INCLUDE COND=(5,4,CH,EQ,C'ABCD')
 SORT FIELDS=(9,3,ZD,A)
 OUTREC FIELDS=(1,80,C'BLANK WITHIN A LITERAL')
```

Note that you should only start with a blank in column 16 of line 2 if you need a blank as the first character of the continued operand, as shown in the OUTREC statement above. A blank in column 16 of line 2 will be included in the operand and will result in invalid syntax if incorrectly placed. For example:

```
*        1         2         3         4         5         6         7
*234567890123456789012345678901234567890123456789012345678901234567890123456789012
                                                    SORT FIELDS=(5,4,Z*
               D,A)
                                                     SUM FIELDS=(5,4,Z*
                D)
```

The above statements will be treated as if they were specified as:

```
 SORT FIELDS=(5,4,ZD,A)
 SUM FIELDS=(5,4,Z D)
```

With the 'D' in column 16 of line 2, we get 'ZD' in the SORT statement. But with the 'D' in column 17 of line 2, we get 'Z D' in the SUM statement instead of 'ZD', resulting in a syntax error.

Explicit continuation in 2-15: If line 1 breaks at column 71 with a nonblank in column 72, and columns 2-15 of line 2 are nonblank, DFSORT continues on line 2 with the first nonblank character it finds in columns 2-15. For example:

```
*        1         2         3         4         5         6         7
*234567890123456789012345678901234567890123456789012345678901234567890123456789012
                                        INCLUDE COND=(5,4,CH,EQ,C'AB*
   CD')
                                                    SORT FIELDS=(9,3,*
    ZD,A)
                                                  OUTREC FIELDS=(5,4,2X*
      ,9,3,ZD,M26,80:X)
```

The above statements will be treated as if they were specified as:

```
 INCLUDE COND=(5,4,CH,EQ,C'ABCD')
 SORT FIELDS=(9,3,ZD,A)
 OUTREC FIELDS=(5,4,2X,9,3,ZD,M26,80:X)
```

Remark continuation in 2-71: If a statement ends on line 1 with a blank before column 72 and a nonblank in column 72, DFSORT treats the first nonblank character it finds in columns 2-71 of line 2 as the start of a remark. For example:

```
*        1         2         3         4         5         6         7
*234567890123456789012345678901234567890123456789012345678901234567890123456789012
   SORT FIELDS=(9,3,ZD,A)                  THIS IS A                   *
                                             CONTINUED REMARK
```

Note that a simpler way to do the same thing (without continuation) is to use a comment statement for line 2. For example:

```
*        1         2         3         4         5         6         7
*234567890123456789012345678901234567890123456789012345678901234567890123456789012
   SORT FIELDS=(9,3,ZD,A)                  THIS IS A
*                                            CONTINUED REMARK
```

DFSPARM Enhancement

When PARMDDN=ddname is specified at installation-time, DFSORT will now use a //DFSPARM DD data set if a //ddname DD data set is not present. When PARMDDN=DFSPARM is specified or defaulted at installation-time, DFSORT will continue to use a //$ORTPARM DD data set if a //DFSPARM DD data set is not present.

General Description

The PARMDDN=ddname installation option indicates the name of the ddname for the DFSPARM data set. DFSORT can now use a //DFSPARM DD statement when PARMDDN=ddname specifies a ddname other than DFSPARM, but a //ddname DD statement is not present. For example, if **PARMDDN=$ORTPARM** is specified, the following variations are possible:

- //$ORTPARM - no //DFSPARM

```
...
//$ORTPARM DD *
* Will be used
  OPTION NOEQUALS
/*
```

- //$ORTPARM and //DFSPARM

```
...
//DFSPARM DD *
* Will not be used because //$ORTPARM DD is present
  OPTION EQUALS
/*
//$ORTPARM DD *
* Will be used
  OPTION NOEQUALS
/*
```

- //DFSPARM - no //$ORTPARM

```
...
//DFSPARM DD *
* Will be used because //$ORTPARM DD is not present
  OPTION EQUALS
/*
```

DFSORT will continue to use a //$ORTPARM DD statement when PARMDDN=DFSPARM is specified or defaulted, but a //DFSPARM DD statement is not present. For example, if **PARMDDN=DFSPARM** is defaulted, the following variations are possible:

- //DFSPARM - no //$ORTPARM

```
...
//DFSPARM DD *
* Will be used
  OPTION EQUALS
/*
```

- //DFSPARM and //$ORTPARM

```
...
//DFSPARM DD *
* Will be used
  OPTION EQUALS
/*
//$ORTPARM DD *
* Will not be used because //DFSPARM DD is present
  OPTION NOEQUALS
/*
```

- //$ORTPARM - no //DFSPARM

```
...
//$ORTPARM DD *
* Will be used because //DFSPARM DD is not present
  OPTION NOEQUALS
/*
```

Option Description

PARMDDN=ddname

Specifies an alternate ddname for the DFSORT DFSPARM data set. If this ddname is present at run time, DFSORT uses it to override parameters from other sources.

ddname specifies a name of 1 to 8 characters. The name must be unique within the job step. Do not use a name that is used by DFSORT (for example, SYSIN).

The default is PARMDDN=DFSPARM.

Notes:

1. When PARMDDN=DFSPARM is specified or defaulted:
   - if a //DFSPARM DD data set is available at run-time, DFSORT will use it
   - if a //DFSPARM DD data set is not available at run-time, DFSORT will use a //$ORTPARM DD data set if available.

   Thus with PARMDDN=DFSPARM, you can choose to specify either a //DFSPARM DD data set or a //$ORTPARM DD data set for a particular DFSORT application.
2. When PARMDDN=ddname is specified:
   - if a //ddname DD data set is available at run-time, DFSORT will use it
   - if a //ddname DD data set is not available at run-time, DFSORT will use a //DFSPARM DD data set if available.

   Thus with PARMDDN=ddname, you can choose to specify either a //ddname DD data set or a //DFSPARM DD data set for a particular DFSORT application.

Changed Messages

This section shows messages that have been changed for PTF UQ90053. Refer to *DFSORT Messages, Codes and Diagnosis Guide* for general information on DFSORT messages.

ICE001A

ICE001A TEXT BEGINS IN WRONG COLUMN

Explanation: Critical. A continuation line started in column 1.

System Action: The program terminates.

Programmer Response: Use one of the following recommended valid methods for continuing a statement:

- Break the previous line at a comma-blank or semicolon-blank or colon-blank before column 72 and start the continuation line in column 2-71, or
- break the previous line at column 71, put a nonblank in column 72 and start the continuation line in column 16.

See *DFSORT Application Programming Guide* for other valid methods of continuing a statement.

ICE016A

ICE016A INVALID FIELDS OPERAND VALUE

Explanation: Critical. The FIELDS operand of a SORT or MERGE statement contained an invalid f or s value in a p,m,s or p,m,f,s field.

System Action: The program terminates.

Programmer Response: Check for invalid f and s values in the control fields.

ICE018A

ICE018A INVALID OR MISSING FORMAT

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

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

```
SORT FIELDS=(5,4,NG,A)          NG IS AN INVALID FORMAT
SUM FIELDS=(12,2),FORMAT=CH     CH IS AN INVALID FORMAT FOR SUM
```

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

```
OUTFIL FNAMES=OUT1,INCLUDE=(5,2,NG,EQ,C'AB')  NG IS AN INVALID FORMAT
OUTFIL FNAMES=OUT2,INCLUDE=(21,2,EQ,35,2)     FORMATS ARE MISSING
```

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

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

System Action: The program terminates.

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

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

For case 5, use an INCLUDE or OMIT statement or change the D2 format to a format that is valid for the INCLUDE or OMIT parameter of an OUTFIL statement.

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

```
OUTFIL FNAMES=OUT1,INCLUDE=(5,2,CH,EQ,C'AB')
OUTFIL FNAMES=OUT2,INCLUDE=(21,2,BI,EQ,35,2,BI)
```

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

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

ICE170I

ICE170I FORMAT OPERAND IGNORED

Explanation: A FORMAT=f operand was specified on a SORT, MERGE, SUM, INCLUDE or OMIT statement in which p,m,f was used for all of the fields. For example:

```
SORT FORMAT=BI,FIELDS=(5,4,CH,A,12,2,PD,A,21,8,BI,D)
```

System Action: f from p,m,f is used for each field. f from FORMAT=f is not used for any of the fields.

Programmer Response: Optional. Remove FORMAT=f, or remove any or all f values in FIELDS that match the f value in FORMAT=f. For example, the SORT statement above could be changed to either of the following SORT statements to eliminate the ICE170I message:

```
SORT FIELDS=(5,4,CH,A,12,2,PD,A,21,8,BI,D)
SORT FORMAT=BI,FIELDS=(5,4,CH,A,12,2,PD,A,21,8,D)
```

ICE189A

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

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

- LOCALE processing
- OUTFIL processing
- Y2x, Y2xx, PD0, FS or CSF format
- INREC or OUTREC processing with one of the following:
  - p,m,HEX
  - p,HEX
  - p,m,TRAN=LTOU
  - p,TRAN=LTOU
  - p,m,TRAN =UTOL
  - p,TRAN=UTOL
  - p,m,TRAN=ALTSEQ
  - p,TRAN=ALTSEQ
  - p,m,f
  - p,m,lookup
  - SEQNUM
  - DATE1, DATE1(c), DATE1P, DATE2, DATE2(c), DATE2P, DATE3, DATE3(c), DATE3P, or DATE4
  - TIME1, TIME1(c), TIME1P, TIME2, TIME2(c), TIME2P, TIME3, or TIME3P
  - +n

  - -n
  - (...)
- A VSAM extended addressability data set
- To set the SORTOUT LRECL from the L3 length (without E35, INREC or OUTREC), the OUTREC length or the INREC length, with SOLRF in effect
- an FL format sort field with NOSZERO in effect
- VLLONG in effect and SORTOUT present
- VSAMEMT in effect for a sort or merge with VSAM input
- The same VSAM data set was specified for both input and output
- An HFS file was specified for input or output
- A tape data set with a block size greater than 32760 bytes was specified for input or output
- SDB=LARGE or SDB=INPUT was in effect and DFSORT selected a block size greater than 32760 bytes for a tape output data set
- VLSHRT in effect with a SUM statement
- Position plus length for a control field exceeded 4093

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

System Action: The program terminates.

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

ICE211I

ICE211I OLD OUTFIL STATEMENT PROCESSING USED

Explanation: This OUTFIL statement did not have any of the following parameters: FNAMES, FILES, STARTREC, ENDREC, SAMPLE, INCLUDE, OMIT, SAVE, OUTREC, VTOF, CONVERT, VLFILL, FTOV, VLTRIM, REPEAT, SPLIT, SPLITBY, LINES, HEADER1, TRAILER1, HEADER2, TRAILER2, SECTIONS, NODETAIL, or REMOVECC. For compatibility, this OUTFIL statement was treated as an "old" OUTFIL statement and all of its parameters were ignored

System Action: Processing continues, but OUTFIL data sets are not associated with this OUTFIL statement. If the Blockset technique is not selected, control statement errors could result from continuation of this OUTFIL statement.

Programmer Response: None, unless this is not an old OUTFIL statement, in which case valid parameters from the list above should be specified.

ICE221A

ICE221A INVALID FIELD OR CONSTANT FOUND IN <ddname> CONDITION n

Explanation: Critical. An error was detected in the COND parameter of an INCLUDE or OMIT statement (ddname is blank), or in the INCLUDE or OMIT parameter of an OUTFIL statement (ddname indicates the first data set in the OUTFIL group). n indicates the number of the relational condition in which one of the following errors was detected:

- the length for a field with a format other than SS was greater than 256.
- the length for a PD field was 256
- the length for a PD0 field was less than 2 or greater than 8
- the length for a CSF field was greater than 16
- the length for a CSL, CST, ASL, or AST field was 1
- the decimal constant for an FI field was greater than 2147483647 or less than -2147483648
- the decimal constant for a BI field was greater than +4294967295 or less than +0
- the number of digits (including leading zeros) in the decimal constant for an FI or BI field was greater than 15
- the length for a Y2 field was not 2 for Y2C, Y2Z, Y2P or Y2S, or 1 for Y2D or Y2B, or 3-6 for Y2T or Y2W, or 2-3 for Y2U or Y2X, or 3-4 for Y2V or Y2Y
- a Y2 field was compared to another Y2 field with a different number of non-year digits
- a Y2 field was compared to a Y constant with a different number of non-year digits
- a Y2 field other than Y2S, Y2T or Y2W was compared to Y'LOW', Y'BLANKS' or Y'HIGH'
- a Y2 field was compared to a decimal constant instead of to a Y constant

System Action: The program terminates.

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

ICE232A

ICE232A ddname: SPLIT, SPLITBY OR REPEAT CANNOT BE USED FOR A REPORT

Explanation: Critical. For the OUTFIL group whose first data set is associated with ddname, a SPLIT, SPLITBY, or REPEAT parameter was specified along with one or more report parameters (LINES, HEADER1, TRAILER1, HEADER2, TRAILER2, SECTIONS or NODETAIL). The records of a report cannot be repeated, or split among a group of OUTFIL data sets.

System Action: The program terminates.

Programmer Response: Remove either the SPLIT, SPLITBY, or REPEAT parameter or the report parameters.

ICE234A

ICE234A STARTREC, ENDREC, SAMPLE OR REPEAT VALUES ARE INCONSISTENT

Explanation: Critical. Specified values were inconsistent in one of the following ways:

- The n value of ENDREC=n is less than the n value of STARTREC=n. Example: STARTREC=10,ENDREC=9.

  The n value of ENDREC=n must be equal to or greater than the n value of STARTREC=n. Example: STARTREC=10,ENDREC=10 to process record 10, or STARTREC=10,ENDREC=20 to process records 10-20.

- The n value of SAMPLE=n or SAMPLE=(n,m) is 1 which would result in processing every record instead of a sample of records. Example: SAMPLE=1.

  The n value of SAMPLE=n or SAMPLE=(n,m) must be equal to or greater than 2. Example: SAMPLE=2 to process records 1, 3, and so on, or SAMPLE=(10,2) to process records 1, 2, 11, 12, and so on.

- The m value is less than or equal to the n value of SAMPLE=(n,m) which would result in processing every record instead of a sample of records. Example: SAMPLE=(10,10).

  The m value must be less than the n value of SAMPLE=(n,m). Example: SAMPLE=(5,4) to process records 1, 2, 3, 4, 6, 7, 8, 9, and so on.

- The n value of REPEAT=n is 1 which would result in no repetitions. Example: REPEAT=1.

  The n value of REPEAT=n must be equal to or greater than 2. Example: REPEAT=5 to repeat each output record five times.

System Action: The program terminates.

Programmer Response: Specify consistent values for STARTREC, ENDREC, SAMPLE and REPEAT.

ICE276A

ICE276A RESERVED WORD - NOT ALLOWED FOR SYMBOL

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

System Action: The program terminates.

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

ICE613A

ICE613A REQUIRED KEYWORD MISSING: keyword

Explanation: Critical. The indicated keyword was required for this operator, but was not specified. The required keywords and their operands for each operator are:

- COPY - FROM, and TO or USING
- COUNT - FROM
- DEFAULTS - LIST
- DISPLAY - FROM, ON, and LIST, and BREAK if BTITLE, BTOTAL, BMAXIMUM, BMINIMUM, or BAVERAGE is specified
- MODE - STOP, CONTINUE, or SCAN
- OCCUR - FROM, LIST, and ON(p,m,f), ON(p,m,HEX), or ON(VLEN)
- RANGE - FROM, ON, and HIGHER, LOWER, EQUAL, or NOTEQUAL
- SELECT- FROM, TO or DISCARD, ON, and ALLDUPS, NODUPS, HIGHER, LOWER, EQUAL, FIRST, LAST, FIRSTDUP, or LASTDUP
- SORT - FROM and USING
- SPLICE - FROM, TO, ON, and WITH
- STATS - FROM and ON
- UNIQUE - FROM and ON
- VERIFY - FROM and ON

System Action: This operation is terminated.

Programmer Response: Supply the indicated keyword or operand.

ICE614A

ICE614A INVALID OPERATOR

Explanation: Critical. The first keyword in the statement was not a valid operator. The valid operators are: COPY, COUNT, DEFAULTS, DISPLAY, MODE, OCCUR (or OCCURS), RANGE, SELECT, SORT, SPLICE, STATS, UNIQUE, and VERIFY.

A common cause of this error is a missing hyphen (-) on the previous line to indicate continuation.

System Action: This operation is terminated.

Programmer Response: A $ marks the point at which the error was detected. If an invalid operator was used, replace it with a valid operator. If this is a continuation line, use a hyphen after the last operand on the previous line.

ICE623A

ICE623A MAXIMUM NUMBER OF keyword KEYWORDS EXCEEDED

Explanation: Critical. Too many keywords of the indicated type were specified for this operator. The maximum number of HEADER fields is 20 for a DISPLAY operator or 10 for an OCCUR operator. The maximum number of WITH operands is 50 for a SPLICE operator. The maximum number of ON fields for each operator is:

- DISPLAY - 20
- OCCUR - 10
- RANGE - 1
- SELECT - 10
- SPLICE - 10
- STATS - 10
- UNIQUE - 1
- VERIFY - 10

System Action: This operation is terminated.

Programmer Response: A $ marks the point at which the error was detected. Reduce the number of indicated keywords for this operator to the maximum allowed. If necessary, use additional operator(s) to handle all the required fields.

ICE624A

ICE624A MAXIMUM NUMBER OF TO DDNAMES EXCEEDED

Explanation: Critical. Too many TO ddnames were specified for this operator. The maximum number of TO ddnames for each operator is:

- COPY - 10
- SELECT - 1
- SORT - 10
- SPLICE - 1

System Action: This operation is terminated.

Programmer Response: A $ marks the point at which the error was detected. Reduce the number of TO ddnames for this operator to the maximum allowed. Use additional operator(s) to handle all the data sets required.

ICE638I

ICE638I NUMBER OF RECORDS RESULTING FROM CRITERIA: nnnnnnnnnnnnnn

Explanation: Indicates the count of records produced as a result of the specified criteria, for example, ALLDUPS (prints as 15 decimal digits padded with zeros on the left as needed).

- For an OCCUR operator, the count indicates the total number of records in the list data set.

- For a SELECT or SPLICE operator, the count indicates the total number of records in the outdd data set.

System Action: None.

Programmer Response: None.