DFSORT: Ask Professor Sort

DFSORT Web Site

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

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

Contents

DFSORT: Ask Professor Sort

Introduction: Details of functions used in answers

For complete information on the DFSORT/ICETOOL functions used in the answers shown here, see:

- DFSORT documentation (September, 2011) at:

  http://www.ibm.com/support/docview.wss?rs=114&uid=isg3T7000080

- "User Guide for DFSORT PTFs UK90025 and UK90026" (October, 2010) at:

  http://www.ibm.com/support/docview.wss?rs=114&uid=isg3T7000242

Note: z/OS DFSORT V1R10 is used for z/OS 1.10 and 1.11. z/OS DFSORT V1R12 is used for z/OS 1.12 and 1.13.

How can I determine my DFSORT PTF function level?

Professor Sort says ...

Many DFSORT functions were delivered as PTFs at various times. You cannot use a specific function delivered via a PTF unless your site has installed that PTF. For example, you can only use DFSORT's IFTRAIL function if your site has installed DFSORT's October, 2010 functional PTF.

Note: z/OS DFSORT V1R10 is used for z/OS 1.10 and 1.11. z/OS DFSORT V1R12 is used for z/OS 1.12 and 1.13.

To determine which level of DFSORT functions you have available, look at messages ICE000I and ICE201I in the //SYSOUT messages you receive from the following DFSORT job, or from any sucessful DFSORT job.

```
//S1 EXEC PGM=ICEMAN
//SYSOUT   DD SYSOUT=*
//SORTIN DD *
RECORD
/*
//SORTOUT DD DUMMY
//SYSIN   DD   *
  OPTION COPY
/*
```

The ICE000I message indicates which version of DFSORT you have.

Note: If you see WERxxxs messages, you have Syncsort, not DFSORT.

If you see the following message, you have z/OS DFSORT V1R12:

```
ICE000I 1 - CONTROL STATEMENTS FOR 5694-A01, Z/OS DFSORT V1R12 ...
```

You are using the latest version of DFSORT and can use new functions introduced with z/OS DFSORT V1R12 such as support for using memory objects as intermediate work space.

If you see the following message, you have z/OS DFSORT V1R10:

```
ICE000I 1 - CONTROL STATEMENTS FOR 5694-A01, Z/OS DFSORT V1R10 ...
```

You cannot use new functions introduced with z/OS DFSORT V1R12.

The ICE201I message indicates which DFSORT functional PTF level you have.

If you see:

```
ICE201I H RECORD TYPE ...
```

the H indicates you have the October, 2010 DFSORT functions (RESIZE, IFTRAIL, ACCEPT, ADDDAYS, DATEDIFF, etc) and all of the earlier functions. This function level corresponds to z/OS DFSORT V1R10 PTF UK90025 and z/OS DFSORT V1R12 PTF UK90026. You are completely up to date on DFSORT functional PTFs.

If you see:

```
ICE201I G RECORD TYPE ...
```

the G indicates you have the November, 2009 DFSORT functions (JOINKEYS, TOJUL, TOGREG, WEEKDAY, etc) and all of the earlier functions. This function level corresponds to z/OS DFSORT V1R10 PTF UK51707. You are behind on DFSORT functional PTFs. Ask your System Programmer to install the October, 2010 PTF.

If you see:

```
ICE201I F RECORD TYPE ...
```

the F indicates you have the July, 2008 DFSORT functions (FINDREP, WHEN=GROUP, DATASORT, SUBSET, etc) and all of the earlier functions. This function level corresponds to z/OS DFSORT V1R10 PTF UK90014. You are behind on DFSORT functional PTFs. Ask your System Programmer to install the October, 2010 PTF.

What level of DFSORT features does the DFSORT website describe?

Professor Sort says ...

All areas of the DFSORT website reflect the features available with z/OS DFSORT V1R10 PTF UK90025 or z/OS DFSORT V1R12 PTF UK90026, and with all earlier PTFs. UK90025 and UK90026 first became available in October, 2010 and are fully documented in *User Guide for DFSORT PTFs UK90025 and UK90026* on the DFSORT website. All features available with all DFSORT PTFs are documented in the z/OS DFSORT V1R13 books (September, 2011).

If you don't have z/OS DFSORT V1R12 (for z/OS 1.12 and 1.13) installed, then features and options available only in that release, will not be available to you:

If you don't have the October, 2010 PTF installed, then features and options available only with that PTF will not be available to you.

Summary of changes by release on the DFSORT website lists new features and options of DFSORT introduced by various releases and PTFs.

What's New in DFSORT on the DFSORT website provides a quick introduction to selected DFSORT and ICETOOL enhancements available as of October, 2010.

Where can I find information to help me use DFSORT more effectively?

Professor Sort says ...

The following are sources of information and examples available from IBM that can help you use DFSORT's many features more effectively.

The DFSORT website

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

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

The DFSORT library

To get the most out of DFSORT, every programmer who uses it should be familiar with the following books in the DFSORT library:

- *DFSORT: z/OS Getting Started (SC26-7527)* is an excellent tutorial about the basics of using DFSORT control statements, ICETOOL operators and Symbols. The material is taught using numerous examples which can be run using sample data sets shipped with DFSORT.
- *z/OS DFSORT Application Programming Guide (SC26-7523)* is a complete reference guide for the functions and options of DFSORT, and includes many examples.
- *z/OS DFSORT Messages, Codes and Diagnosis Guide (SC26-7525)* can help you eliminate common sources of errors, interpret informational (I) and error (A) messages, and diagnose program failures.
- *z/OS DFSORT Tuning Guide (SC26-7526)* provides valuable information about tuning DFSORT and using it for effective Application Development.

Programmers who install DFSORT should use *z/OS DFSORT Installation and Customization (SC26-7524)* as a reference source.

You can access all of the DFSORT books online from the DFSORT website.

Online DFSORT books

Online books provide capabilities (for example, search) not provided by traditional books. All of the DFSORT books can be accessed online from the "Publications" link on the DFSORT website. Bookmark this link for faster access.

LookAt

You can use IBM's LookAt facility at:

http://www.ibm.com/systems/z/os/zos/bkserv/lookat/index.html

to access information on a specific DFSORT message by typing in its message number (e.g. ICE283A).

Other IBM books

The following are some of the IBM books that contain information on using DFSORT and ICETOOL with other IBM products:

- *z/OS DFSMSrmm Reporting (SC26-7406)*
- *z/OS DFSMShsm Data Recovery Scenarios (GC35-0419)*
- *z/OS DFSMShsm Storage Administration Guide* (SC35-0421)
- *z/OS SecureWay Security Server RACF Auditor's Guide (SA22-7684)*
- *z/OS SecureWay Security Server RACF Security Administrator's Guide (SA22-7683)*
- *z/OS SecureWay Security Server RACF System Programmer's Guide (SA22-7681)*
- *z/OS MVS System Management Facilities (SMF) (SA22-7630)*
- *z/OS DFSMS Introduction (SC26-7397)*
- *z/OS DFSMS Access Method Services for Catalogs (SC26-7394)*
- *z/OS DFSMS: Using the Volume Mount Analyzer (SC26-7413)*
- *z/OS DCE Administration Guide (SC24-5904)*
- *z/OS Language Environment Programming Guide (SA22-7561)*
- *DB2 Universal Database for OS/390 and z/OS Utility Guide and Reference (SC26-9945)*

Papers and examples

The DFSORT website contains various papers that can help programmers be more productive with DFSORT and ICETOOL.

Where can I ask specific questions about DFSORT?

Professor Sort says ...

If you think you've found a defect in DFSORT, you should contact IBM TotalStorage Technical Support at:

http://www.ibm.com/servers/storage/support/

If you have a performance or migration question about DFSORT, you can contact the DFSORT Hotline at the following e-mail address:

dfsort@us.ibm.com

This is a direct line to the DFSORT Team and they'll do their best to respond as quickly as possible.

If you have a "how-to" question about DFSORT, here are some good places where you can ask for help:

- IBMMAINFRAMES at:

  http://ibmmainframes.com/index.php

Post your question on the IBMMAINFRAMES Help Board in the "DFSORT/ICETOOL" topic. (You'll need to register the first time before you can post.) The DFSORT Team monitors this topic and responds when appropriate along with other help board participants.

- MVSFORUMS at:

  http://www.mvsforums.com

  Post your question on the MVSFORUMS Help Board in the "Utilities" topic. (You'll need to register the first time before you can post.) The DFSORT Team monitors this topic and responds when appropriate along with other help board participants.

- IBM-MAIN at:

  http://bama.ua.edu/archives/ibm-main.html

  Post your question on this newsgroup/mailing list. The DFSORT Team monitors this list and responds when appropriate along with other newsgroup/mailing list participants.

Can I set DFSORT installation options from PARMLIB members?

Professor Sort says ...

You can set DFSORT installation options using ICEPRMxx members in concatenated PARMLIB. Each ICEPRMxx member can contain options to be changed for any or all of DFSORT's eight installation environments (JCL, INV, TSO, TSOINV, TD1, TD2, TD3 and TD4). Up to ten ICEPRMxx members can be activated by a START ICEOPT started task command. The options in the activated members will be merged with the ICEMAC defaults at run-time.

See "How can I let DFSORT use more resources on weekends and off-shift?" in this paper for an example of using an ICEPRMxx member. See *DFSORT Installation and Customization* for complete details on using ICEPRMxx members.

ICEPRMxx members are the recommended way to change DFSORT installation defaults.

How can I let DFSORT use more resources on weekends and off-shift?

Professor Sort says ...

DFSORT's time-of-day options control feature makes this easy to do.

You're probably familiar with the JCL, INV, TSO and TSOINV parameters that let you set installation options for four different invocation environments. Well, DFSORT goes a step further and offers the TD1-TD4 parameters that let you set installation options for four different time-of-day envioronments. You can use any or all of the time-of-day environments for any or all of the invocation environments.

You can use ICEPRMxx members in PARMLIB (recommended) to override installation options for any or all of the four invocation environments and four time-of-day environments. (Alternatively, you can use the ICEMAC macro in a usermod to override the installation options.)

So, for example, you could use an ICEPRM01 member of PARMLIB to raise the DSA value from the default value of 64 (megabytes) to a larger value of 96 (megabytes) for DFSORT jobs that start off-shift and on weekends. Here's how:

```
ICEPRM01 member
---------------
JCL
  ENABLE=TD1
  SVC=(,ALT)
INV
  ENABLE=TD1
  SVC=(,ALT)
TD1
  WKDAYS=(1800,559)
  WKEND=ALL
  SVC=(,ALT)
  DSA=96
```

You would activate the ICEPRM01 member with this START ICEOPT command:

```
START ICEOPT,ICEPRM=01
```

DFSORT batch jobs (JCL, INV) that start on Monday-Friday (WKDAYS) between 6:00pm (1800) and 5:59am (559) or on weekends (WKEND=ALL) will use the TD1 installation defaults (for example, DSA=96) instead of the JCL or INV defaults (for example, the shipped default of DSA=64).

By setting your JCL, INV, TSO, TSOINV and TD1-TD2 defaults appropriately, you can fine-tune DFSORT's resource usage for special situations at your site. For complete information on the four invocation environments and the four time-of-day environments, see *z/OS DFSORT Installation and Customization.*

What are the advantages of dynalloc over SORTWKdd DDs?

Professor Sort says ...

Dynamic allocation of work data sets has the following advantages over JCL allocation:

- As the characteristics (for example, file size and virtual storage size) of an application change over time, DFSORT can automatically optimize the amount of dynamically allocated work space for the application. This reduces unneeded allocation of DASD space.
- As the amount of central storage available to the application varies from run to run, DFSORT can automatically adjust the amount of space it dynamically allocates to complement the amount of central storage to be used for sorting in memory. This reduces unneeded allocation of DASD space.
- The amount of work space actually used is often less than the amount allocated. DFSORT tries to minimize dynamic over-allocation while making certain that the application does not fail due to lack of space. With JCL allocation, you could minimize the amount of allocated space manually, but this might require changes to JCL allocation as the characteristics of the application change.
- DFSORT's use of dynamic allocation has logic built in that allows it to attempt recovery when additional work space is required.
- With dynamic allocation, DFSORT also allocates work data sets as large format, allowing a single work data set to use more than 64K tracks on a single volume.

You set the DYNAUTO installation option to control whether dynamic allocation is used automatically, or only when requested by the DYNALLOC run-time option. DYNAUTO also controls whether dynamic allocation or JCL allocation takes precedence when JCL work data sets are specified.

DYNAUTO can be set as follows:

- If DYNAUTO=IGNWKDD, dynamic allocation takes precedence over JCL allocation. If you want the opposite result for selected applications, use the USEWKDD run-time option.
- If DYNAUTO=YES, JCL allocation takes precedence over dynamic allocation. If you want the opposite result, remove all JCL allocation statements.
- If DYNAUTO=NO, dynamic allocation of work data sets is not used unless you specify the DYNALLOC run-time option. JCL allocation takes precedence over dynamic allocation.

Here are some additional things to consider for JCL allocation of work data sets vs dynamic allocation of work data sets:

Performance

There's little difference in performance between using JCL allocation or dynamic allocation. The benefits of using dynamic allocation are more related to reliability and efficient use of disk space. We have many customers sorting extremely large files using DFSORT's dynamic allocation to obtain the required disk work space.

Efficient use of disk work space

With dynamic allocation, DFSORT calculates the disk work space required at the time a sort executes. So as file sizes increase or decrease, the disk allocation is calculated accordingly. If DFSORT plans to use Hiperspace, Memory Objects or Dataspace, it can reduce the disk allocation based on the expected central storage usage. With JCL allocation, you would have to adjust the JCL space allocations manually as the file sizes grow. But there's really no way you could know in advance how much Hiperspace, Memory Object or Dataspace a sort is going to use because that would depend on available resources at the time the sort executes.

Reliability

DFSORT's use of dynamic allocation has logic built in that allows it to attempt recovery when additional work space is required. This can happen in cases where more records are sorted than were expected (for example, if an E15 inserts a large number of records). If JCL allocation is used instead of dynamic allocation, DFSORT cannot take those recovery actions.

With dynamic allocation, DFSORT also allocates work data sets as large format, allowing a single work data set to use more than 64K tracks on a single volume. With JCL allocation, you would have to remember to use DSNTYPE=LARGE on your SORTWKnn DD statements in order to take advantage of large format work data sets.

Maintainability

As mentioned above, with dynamic allocation, DFSORT allocates the correct amount of space based on the file size and expected storage usage. With JCL allocation, you would have to periodically review those space allocations to determine if they are accurate for the size of files being sorted. Alternatively, you could just wait until a sort starts to fail with a Sort Capacity Exceeded message (ICE046A) and then increase the JCL allocation.

In addition:

While DFSORT can dynamically calculate the amount of space required, it cannot dynamically change the number of work data sets to use. The shipped default for the DYNALOC installation default is (SYSDA,4) which causes

only 4 work data sets to be used. So for a 32GB sort, you would need at least 4 volumes each with 8GB of free space. (Actually, you'd need a bit more than that due to "overhead".) If 4 volumes with that much free space are not available, DFSORT would fail with a RESOURCES UNAVAILABLE FOR DYNAMIC ALLOCATION message (ICE083A).

For most shops, we recommend setting the default DYNALOC number to a value that accommodates the majority of sorts. For extremely large sorts, you can then override that default value by passing a larger value with the DYNALLOC parameter. So for that 32GB sort you might want to pass something like DYNALLOC=(SYSDA,16) so the work space allocation is spread across more volumes. The total space allocated is the same, but DFSORT doesn't need as much free space on a single volume. Yes, you might have to tweak that number if there's a big change in the file size over time, but not nearly as frequently as you'd have to adjust JCL allocations.

Another concern for using dynamic allocation is that it requires accurate file size information. For JCL invoked sorts with input on disk, DFSORT is able to calculate the file size automatically. But in the following situations, DFSORT might need some help determining an accurate file size:

1. All records are passed to DFSORT (inserted) by an E15 exit. DFSORT has no way of knowing the number of records to be inserted by the exit. If the inserted records are variable length, DFSORT doesn't know the average record length either.
2. Input is from disk but an E15 exit inserts additional records. DFSORT has no way of knowing the number of records to be inserted by the exit.
3. Input is from tape and a tape management system does not provide file size information to DFSORT. DFSORT provides an interface for obtaining file size information from tape management systems like RMM.

1, 2 and 3 can be addressed by providing FILSZ and/or AVGRLEN values to DFSORT, as appropriate. Obviously these values might need to be adjusted as the amount of data to be sorted increases or decreases over time, but you'd have to adjust JCL allocations as well. Dynamic allocation also has the advantage of allowing DFSORT to add some buffer in its calculations for when the file size is larger than expected.

DFSORT also provides a DYNSPC operand which is the amount of work space to dynamically allocate when the file size is unknown. If you feel you have a number of sorts with unknown file size (indicated by message ICE118I), then you could increase DYNSPC from the shipped default of 256MB to something much larger.

To further improve reliability with dynamic allocation, DFSORT introduced the DYNAPCT operand in DFSORT V1R12 to allow allocation of additional work data sets to be allocated with zero space. DFSORT only extends these data sets when necessary to complete a sort application. The DYNAPCT operand specifies the number of additional work data sets as a percentage of the DYNALOC/DYNALLOC 'n' value at run time. If you feel you have a number of sorts that provide unreliable file size information, then you could increase DYNAPCT from the shipped default of 10% to something larger.

Additional sources of information can be found in the following DFSORT publications available from the DFSORT website

- *DFSORT Application Programming Guide:* Describes the FILSZ and AVGRLEN operands. Also see the chapter on Improving Efficiency.
- *DFSORT Installation and Customization:* Describes the DYNSPC operand, and the There's also a chapter on improving tape processing with RMM ICETPEX exit for improving tape processing with tape management systems.
- *DFSORT Tuning Guide:* Discusses dynamic allocation of work Data sets.

How can I list my site's DFSORT installation defaults?

Professor Sort says ...

With DFSORT's ICETOOL utility, you can produce a report listing your site's current DFSORT installation defaults any time you want to.

Here's an example of a complete ICETOOL job to do it. It uses the DEFAULTS operator to produce the report:

```
//DEFAULTS  JOB  <job Card Parameters>
//SHOWDEF   EXEC PGM=ICETOOL
//TOOLMSG   DD   SYSOUT=A
//DFSMSG    DD   SYSOUT=A
//LIST1     DD   SYSOUT=A
//TOOLIN    DD   *
  DEFAULTS LIST(LIST1)
/*
```

DEFAULTS produces a three-part report showing:

1. The merged PARMLIB/ICEMAC installation default values for ICEAM1-4 and ICETD1-4 that will be used at run-time. If the installation default value for an item is different from the IBM-supplied default value, the IBM-supplied default value is shown below the installation default value.
2. The specified PARMLIB ICEPRMxx member option values for ICEAM1-4 and ICETD1-4 (for reference).
3. The ICEMAC installation default values for ICEAM1-4 and ICETD1-4 (for reference).

See *z/OS DFSORT Installation and Customization* for an example of a DEFAULTS report.

How can I supply control statements for a program that calls DFSORT?

Professor Sort says ...

You can use a **DFSPARM data set** to pass control statements and EXEC parameters to DFSORT regardless of whether it's called directly (PGM=ICEMAN) or from a program like COBOL, PL/I or ICETOOL. A DFSPARM data set can be used to override or add control statements and parameters from any other source such as SYSIN, SORTCNTL, EXEC PARM or a calling program's parameter list.

By default, DFSORT uses either a **DFSPARM DD statement** or a **$ORTPARM DD statement** (in that order) as the DFSPARM data set. However, you can change the ddname of the DFSPARM data set with installation option PARMDDN=ddname.

Here's an example of a DFSPARM data set that supplies both control statements and EXEC parameters:

```
//DFSPARM DD *
ABEND,STOPAFT=100
  SORT FIELDS=(5,20,CH,A)
  OMIT COND=(35,2,PD,GT,+20)
  OPTION SORTIN=DATAIN
/*
```

What should I know about migrating to DFSORT from other sort products?

Professor Sort says ...

If you are migrating to DFSORT from another sort product, make sure you have z/OS DFSORT V1R12 (for z/OS 1.12 and z/OS 1.13) or z/OS DFSORT V1R10 (for z/OS 1.10 and z/OS 1.11) with the latest functional PTFs installed. In particular, make sure you have z/OS DFSORT V1R10 PTF UK90025 or z/OS DFSORT V1R12 PTF UK90026 (October, 2010) installed since this PTF provides new options and features that make migration easier.

DFSORT can use information it obtains from a tape management system to significantly improve the way it processes tape data sets. However, certain requirements must be met to enable DFSORT to do this, depending on the tape management system you use. See "Can DFSORT obtain information from my tape management system?" in this paper for details.

If you have specific migration questions, you can contact the DFSORT Hotline at the following e-mail address:

dfsort@us.ibm.com

Many of the new and previously available migration features incorporated into DFSORT make it operate like other sort products automatically. However, the options shown in the Table below have IBM-supplied installation defaults, as indicated, that you may want to change to make DFSORT operate more like the sort product you are migrating from. In particular, consider setting **DYNALOC=(SYSDA,8)** to reduce differences in dynamic allocation of work space.

You can use ICEPRMxx members in PARMLIB to change your DFSORT installation options. See *z/OS DFSORT Installation and Customization* for complete details.

Changing an installation option changes the way DFSORT works globally by default. You can find complete details on all of DFSORT's installation options in *z/OS DFSORT Installation and Customization.*

Specifying a run-time option changes the way DFSORT works for a specific application. You can find complete details on all of DFSORT's run-time options in *z.OS DFSORT Application Programming Guide.*

For general information on DFSORT, you can access all of the DFSORT books online from the DFSORT website.

Table 1 (Page 1 of 2). Table of Options that can make migration easier

Installation option	Run-time option	Specifies ...
ABCODE=MSG ABCODE=n Default: MSG		The ABEND code for a critical error.
DYNALOC=(d,n) Default: SYSDA,4	DYNALLOC=(d,n)	The device name and maximum number of dynamically allocated work data sets.
DYNAUTO=YES DYNAUTO=IGNWKDD DYNAUTO=NO Default: YES	DYNALLOC=(d,n)	whether work data sets are dynamically allocated.
DYNSPC=n Default: 256	DYNSPC=n	primary space (in megabytes) for dynamically allocated work data sets when the file size is unknown.

Table 1 (Page 2 of 2). Table of Options that can make migration easier

Installation option	Run-time option	Specifies ...
EQUALS=YES EQUALS=NO Default: VLBLKSET	EQUALS NOEQUALS	whether the order of records that collate identically is preserved from input to output.
EXITCK=STRONG EXITCK=WEAK Default: STRONG	EXITCK=STRONG EXITCK=WEAK	whether DFSORT terminates or continues for invalid return codes from E15/E35 user exits.
FSZEST=YES FSZEST=NO Default: NO	FILSZ=n FILSZ=En FILSZ=Un	whether DFSORT treats file sizes as exact or estimated.
NOMSGDD=QUIT NOMSGDD=ALL NOMSGDD=CRITICAL NOMSGDD=NONE Default: QUIT		whether DFSORT terminates or continues when the message data set is not available.
PARMDDN=ddname Default: DFSPARM		an alternate ddname, such as $ORTPARM, for the DFSPARM control data set.
RESET=YES RESET=NO Default: YES	RESET NORESET	whether DFSORT processes a VSAM set defined with REUSE as NEW or MOD.
SORTLIB=SYSTEM SORTLIB=PRIVATE Default: PRIVATE		whether DFSORT searches a system or private library for tape work data set sort or Conventional merge modules.
SZERO=YES SZERO=NO Default: YES	SZERO NOSZERO	whether DFSORT treats zero values as signed or unsigned.
VLLONG=YES VLLONG=NO Default: NO	VLLONG NOVLLONG	whether DFSORT truncates long variable-length output records.
VLSCMP=YES VLSCMP=NO Default: NO	VLSCMP NOVLSCMP	whether DFSORT pads short variable-length compare fields.
VSAMEMT=YES VSAMEMT=NO Default: YES	VSAMEMT NVSAMEMT	whether DFSORT accepts an empty VSAM input data set.
VSAMIO=YES VSAMIO=NO Default: NO	VSAMIO NOVSAMIO	whether DFSORT allows a VSAM data set defined with REUSE to be sorted in-place.
ZDPRINT=YES ZDPRINT=NO Default: YES	ZDPRINT NZDPRINT	whether DFSORT produces printable numbers from positive summed ZD fields.

What are the equivalent DFSORT formats for various COBOL data types?

Professor Sort says ...

Both DFSORT and COBOL support a large number of data types. COBOL describes these data types in one way, and DFSORT describes them in another way. If you SORT or MERGE with COBOL, the compiler automatically generates a SORT or MERGE control statement for you with the correct DFSORT descriptions for the COBOL fields you specify. But to take full advantage of DFSORT, you will often want to describe your fields in your own DFSORT control statements (e.g. SORT, MERGE, INCLUDE, OMIT, INREC, OUTREC, OUTFIL, SUM) either outside of COBOL or in a DFSPARM data set used with COBOL. The table below will show you what DFSORT length and format to use for the various commonly used COBOL data types.

For example, say you want to separate out records in a very large file into two data sets based on the values in a PIC S9(4) COMP field starting in position 21. In the first data set, you want records with values in the field that are greater than or equal to +5000. In the second data set, you want records with values in the field that are less than -1000. You could use the table below to determine that a PIC S9(4) COMP field is equivalent to a DFSORT field with a length of 2 and a format of FI, allowing you to code your DFSORT statements as follows:

```
OPTION COPY
OUTFIL FNAMES=OUT1,INCLUDE=(21,2,FI,GE,+5000)
OUTFIL FNAMES=OUT2,INCLUDE=(21,2,FI,LT,-1000)
```

Table 2. Equivalent DFSORT formats for various COBOL data types

COBOL data type	DFSORT length	DFSORT format
PIC X(n) USAGE DISPLAY	n	CH
GROUP DATA ITEMS with n bytes	n	CH
PIC 9(n) DISPLAY	n	ZD
PIC S9(n) DISPLAY <TRAILING>	n	ZD
PIC S9(n) DISPLAY LEADING	n	CLO
PIC S9(n) DISPLAY SEPARATE <TRAILING>	n+1	CST
PIC S9(n) DISPLAY LEADING SEPARATE	n+1	CSL or FS
PIC 9(n) COMP|BINARY|COMP-4|COMP-5		
n = 1 to 4	2	BI
n = 5 to 9	4	BI
n >= 10	8	BI
PIC S9(n) COMP|BINARY|COMP-4|COMP-5		
n = 1 to 4	2	FI
n = 5 to 9	4	FI
n >= 10	8	FI
PIC 9(n) COMP-3|PACKED-DECIMAL	(n/2)+1	PD
PIC S9(n) COMP-3|PACKED-DECIMAL	(n/2)+1	PD
COMP-1	4	FL
COMP-2	8	FL

Notes:

1. PIC 9(x)V9(y) can be treated like PIC 9(n) where n=x+y. (COBOL does NOT store the decimal point internally.)
2. PIC S9(x)V9(y) can be treated like PIC S9(n) where n=x+y. (COBOL does NOT store the decimal point internally.

Can DFSORT obtain information from my tape management system?

Professor Sort says ...

DFSORT can use information it obtains from a tape management system to significantly improve the way it processes tape data sets. However, certain requirements must be met to enable DFSORT to do this, depending on the tape management system you use. Below are the details.

DFSMSrmm

For tapes managed by DFSMSrmm:

- DFSORT can **automatically** obtain accurate input file size information for DFSMSrmm managed tapes. This can result in improved sort performance and more accurate dynamic workspace allocation.

  Additionally, you don't have to supply the input file size to DFSORT when this information is available from DFSMSrmm. DFSORT will automatically use the file size it obtains from DFSMSrmm to override any FILSZ=En or SIZE=En value you specify. However, you must remove any FILSZ=n, FILSZ=Un, SIZE=n or SIZE=Un value you specify in order for DFSORT to use the file size it obtains from DFSMSrmm.
- DFSORT can **automatically** obtain input and output attributes such as RECFM, LRECL and BLKSIZE for DFSMSrmm managed tapes. As a result, you don't have to specify these attributes explicitly for input and output tape data sets when this information is available from DFSMSrmm.

Other tape management systems

For tapes managed by tape management systems other than DFSMSrmm, DFSORT supports a tape exit that allows any tape management system to improve the way it interfaces with DFSORT.

Your tape management system vendor can write an **ICETPEX routine** to supply the same information to DFSORT that it gets automatically from DFSMSrmm. Ask your vendor if they currently have an ICETPEX routine available or have plans to provide one in the future.

An ICETPEX routine can pass specific information to DFSORT for managed input and output tapes. DFSORT can use this information, when appropriate, to significantly improve the way it processes managed tapes:

- DFSORT can **automatically** obtain accurate input file size information for managed tapes from ICETPEX. This can result in improved sort performance and more accurate dynamic workspace allocation.

  Additionally, you don't have to supply the input file size to DFSORT when this information is available from ICETPEX. DFSORT will automatically use the file size it obtains from ICETPEX to override any FILSZ=En or SIZE=En value you specify. However, you must remove any FILSZ=n, FILSZ=Un, SIZE=n or SIZE=Un value you specify in order for DFSORT to use the file size it obtains from ICETPEX.

- DFSORT can **automatically** obtain input and output attribute such as RECFM, LRECL and BLKSIZE for managed tapes from ICETPEX. As a result, you don't have to specify these attributes explicitly for input and output tape data sets when this information is available from ICETPEX.

If your vendor supplies an ICETPEX routine, just include it in the library that contains the DFSORT modules or in a separate library in the normal order of search. DFSORT will automatically call the ICETPEX routine and use the information it provides, when appropriate.

How does DFSORT take advantage of central storage?

Professor Sort says ...

DFSORT can exploit central storage by:

- using Hiperspaces as a way to store intermediate data in central storage instead of using disk work data sets. A Hiperspace is a range of up to two gigabytes of contiguous virtual storage addresses that a program can use as a buffer. While the size of a Hiperspace is limited to 2 gigabytes, it can be backed by real storage above 2 gigagytes. Hipersorting applications can create up to 16 Hiperspaces to store up to 32 gigabytes of intermediate data.
- using memory objects as a way to store intermediate data in central storage instead of using disk work data sets. A memory object is a data area in virtual storage that is allocated above the bar and backed by central storage. Since memory objects are allocated above the bar, they can be much larger than 2 gigabytes. DFSORT is able to store up to 4 gigabytes of intermediate data in a memory object. Similar to Hipersorting, DFSORT can create up to 16 memory objects to store up to 64 gigabytes of intermediate data.
- using a memory object as main storage for sort applications, when appropriate. Memory object sorting is a DFSORT capability that exploits above the bar storage to improve the performance of sort applications. With memory object sorting, DFSORT can create one memory object that can be used exclusively, or along with disk space, as a temporary storage area for sorting records. DFSORT has been successfully tested with memory objects up to 250GB, but even larger memory objects could be used.
- using a data space as main storage for sort applications, when appropriate. A data space is a range of up to two gigabytes of contiguous virtual storage addresses that a program can directly manipulate through assembler instructions. While the size of a data space is limited to 2 gigabytes, it can be backed by real storage above 2 gigagytes. With data space sorting, DFSORT can create one data space that can be used exclusively, or along with disk space, as a temporary storage area for sorting records.

DFSORT chooses the best method for exploiting central storage for each sort application based on available resources, input file characteristics, DFSORT control statements and DFSORT installation defaults. Since all applications can use central storage, contention for this resource must be closely monitored and controlled. Informational APAR II13495 contains additional information and considerations related to DFSORT's use of storage in 64-bit environments with recommendations for tailoring DFSORT's installations defaults to control its use of central storage.

How can memory objects, data spaces and Hiperspaces be customized?

Professor Sort says ...

DFSORT provides installation options that can be used to control the amount of central storage a single sorting application can use:

- MOSIZE controls the amount of memory object storage a single sort application can use. The shipped default for MOSIZE is MAX.
- DSPSIZE controls the amount of data space a single sort application can use. The shipped default for DSPSIZE is MAX.
- HIPRMAX controls the amount of Hiperspace a single sort application can use. The shipped default for HIPRMAX is OPTIMAL.

The EXPMAX, EXPOLD and EXPRES installation options can be used to control the total storage used for:

- Memory object sorting applications when MOSIZE is not 0.
- Data space sorting applications when DSPSIZE is not 0.
- Hipersorting applications when HIPRMAX is not 0.

When evaluating available resources, DFSORT considers two types of central storage frames:

- Available frames are those frames not assigned to an active user
- Old frames are those frames that are assigned to an active user but have not recently been referenced making them eligible for page stealing.

EXPMAX allows you to specify the maximum total amount of available and old central storage to be used at any one time by all Hipersorting, memory object sorting and dataspace sorting applications .

EXPOLD allows you to specify the maximum total amount of old central storage to be used at any one time by all Hipersorting, memory object sorting and dataspace sorting applications.

EXPRES allows you to specify the minimum amount of available and old central storage to be reserved for use by non-Hipersorting, non-memory object sorting, and non-dataspace sorting applications.

We recommend that MOSIZE, DSPSIZE and HIPRMAX be kept as the defaults. This allows DFSORT to control the use of these sorting in memory techniques based on available resources and total DFSORT usage as controlled by the EXPMAX, EXPOLD and EXPRES installation defaults.

How can I reformat my records?

Professor Sort says ...

INREC, OUTREC and OUTFIL let you reformat your records in a variety of ways by adding, deleting or changing various fixed length and variable length (delimited) fields, by finding and replacing or removing constants, and by performing various group operations. You can use any combination of separation fields, edited and unedited input fields, edited decimal constants, edited results of arithmetic expressions, sequence numbers, replaced constants, removed constants, propagated fields, group identifiers, and group sequence numbers.

You can use **INREC** to reformat your records **before** they are sorted, copied or merged.

You can use **OUTREC** to reformat your records **after** they are sorted, copied or merged.

You can use **OUTFIL** to reformat your records **after** they are sorted, copied or merged while creating multiple output data sets and reports.

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

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

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

  Note: You can use FIELDS instead of BUILD for INREC or OUTREC. You can use OUTREC instead of BUILD for OUTFIL.

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

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

- **FINDREP:** Reformat each record by doing various types of find and replace operations. Example:

```
INREC FINDREP=(IN=C'Mr.',OUT=C'Mister')
```

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

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

You can choose to include any or all of the following items in your reformatted INREC, OUTREC or OUTFIL records:

- Fixed position/length fields or variable position/length fields. For fixed fields, you specify the starting position and length of the field directly. For variable fields, such as delimited fields, comma separated values (CSV), tab separated values, blank separated values, keyword separated fields, null-terminated strings (and many other types), you define rules that allow DFSORT to extract the relevant data into fixed parsed fields, and then use the parsed fields as you would use fixed fields.
- Blanks, binary zeros, character strings, and hexadecimal strings
- Current date, future date, past date, and current time in various forms
- Unedited input fields aligned on byte, halfword, fullword, and doubleword boundaries
- Replaced or removed strings.
- Hexadecimal or bit representations of binary input fields
- Characters translated from uppercase to lowercase, lowercase to uppercase, ASCII to EBCDIC or EBCDIC to ASCII.
- Left-justified, right-justified, left-squeezed, or right-squeezed input fields.
- Numeric input fields of various formats converted to different numeric formats, or to character format edited to contain signs, thousands separators, decimal points, leading zeros or no leading zeros, and so on.
- Decimal constants converted to different numeric formats, or to character format edited to contain signs, thousands separators, decimal points, leading zeros or no leading zeros, and so on.
- The results of arithmetic expressions combining fields, decimal constants, operators (MIN, MAX, MUL, DIV, MOD, ADD and SUB) and parentheses converted to different numeric formats, or to character format edited to contain signs, thousands separators, decimal points, leading zeros or no leading zeros, and so on.

- SMF, TOD and ETOD date and time fields converted to different numeric formats, or to character format edited to contain separators, leading zeros or no leading zeros, and so on.
- Input date fields of one type (CH, ZD, PD, 2-digit year, 4-digit year, Julian, Gregorian) converted to corresponding output date fields of another type or to a corresponding day of the week.
- The results of various types of arithmetic operations for input date fields.
- Sequence numbers in various formats.
- A character constant, hexadecimal constant or input field selected from a lookup table, based on a character, hexadecimal or bit constant as input.
- A zoned decimal group identifier, a zoned decimal group sequence number, or a field propagated from the first record of a group to all of the records of a group.

What is JOINKEYS and how can it help me?

Professor Sort says ...

JOINKEYS is a powerful DFSORT function that helps you to perform various "join" and "match" applications on two data sets by one or more keys. You can do an inner join, full outer join, left outer join, right outer join and unpaired combinations. The two data sets can be of different types (fixed, variable, VSAM, and so on) and lengths, and have keys in different locations. You can even do cartesian joins.

For added flexibility, the records from the input data sets can be processed in a variety of ways before and after they are joined using other DFSORT control statements such as INCLUDE, OMIT, INREC, OUTREC, SUM and OUTFIL.

By using DFSORT's JOINKEYS, JOIN and REFORMAT statements in various ways, you can do many different types of join and match operations to "slice and dice" your data.

What is ICETOOL and how can it help me?

Professor Sort says ...

ICETOOL is a versatile data set processing and reporting utility that provides an easy-to-use batch front-end for DFSORT. ICETOOL combines new features with previously available DFSORT features to perform complex sorting, copying, reporting and analytical tasks using multiple data sets in a single job step.

The 17 ICETOOL operators briefly described below provide new tools for programmers:

- COPY - copies a data set to one or more output data sets. Multiple output is handled using a single pass over the input.
- COUNT - prints a message containing the count of records in a data set. Can also be used to create an output data set containing text and the count, or to set RC=12, RC=8, RC=4, or RC=0 based on the count of records in a data set (that is, empty, not empty, higher, lower, equal or not equal).
- DATASORT - sorts data records between header and trailer records in a data set to an output data set.
- DEFAULTS - prints the DFSORT installation defaults in a separate list data set.
- DISPLAY - prints the values and characters of specified numeric and character fields in a separate list data set. Simple, tailored or sectioned reports can be produced. Maximums, minimums, totals, averages and counts can be produced.

- MERGE - merges one or more data set to one or more output data sets. Multiple output is handled using a single pass over the input.
- MODE - sets/resets scanning and error actions.
- OCCUR - prints each unique value for specified numeric (including SMF date/time) and character fields, and the number of times it occurs, in a separate list data set. Simple or tailored reports can be produced. The values printed can be limited to those for which the value count meets specified criteria (that is, all duplicates, no duplicates, higher, lower or equal).
- RANGE - prints a message containing the count of values in a range (that is, higher, lower, equal or not equal) for a numeric field
- RESIZE - creates a larger record from multiple shorter records, or creates multiple shorter records from a larger record, that is, resizes fixed length records.
- SELECT - selects records for an output data set based on meeting criteria (that is, all duplicates, no duplicates, first, first n, last, first duplicate, first n duplicates, last duplicate, higher, lower or equal) for the number of times numeric or character field values occur. Records that are not selected can be saved in a separate output data set
- SORT - sorts a data set to one or more output data sets. Multiple output is handled using a single pass over the input.
- SPLICE - splices together fields from records that have the same numeric or character field values (that is, duplicate values), but different information. Fields from two or more records can be combined to create an output record. The fields to be spliced can originate from records in different data sets, so you can use SPLICE to do various "join" and "match" operations.
- STATS - prints messages containing the minimum, maximum, average, and total of values in numeric fields.
- SUBSET - selects records from a data set based on keeping or removing header records (the first n records), relative records, or trailer records (the last n records). Records that are not selected can be saved in a separate output data set
- UNIQUE - prints a message containing the count of unique values in a numeric or character field.
- VERIFY - prints a message identifying each invalid value found in decimal fields.

Here's an example of the JCL and control statements for an ICETOOL job. Other ICETOOL examples can be found in *z/OS DFSORT Application Programming Guide*, *z/OS DFSORT: Getting Started*, and in *ICETOOL mini-user guide* on the DFSORT website.

```
//EXAMP    JOB   A492,PROGRAMMER
//TOOL     EXEC  PGM=ICETOOL
//TOOLMSG  DD    SYSOUT=A
//DFSMSG   DD    SYSOUT=A
//TOOLIN   DD    *
* Statistics from all branches
STATS FROM(ALL) ON(18,4,ZD) ON(28,6,PD) ON(22,6,PD)
* Books from VALD and WETH
SORT FROM(BKS) TO(DAPUBS,PRPUBS) USING(SPUB)
* Separate output for California and Colorado branches
SORT FROM(ALL) USING(CACO)
* California branches profit analysis
  RANGE FROM(CADASD) ON(28,6,PD) HIGHER(-1500) LOWER(+8000)
* Branches with less than 32 employees
  RANGE FROM(ALL) ON(18,4,ZD) LOWER(32)
* Print profit, employees, and city for each Colorado branch
DISPLAY FROM(CODASD) LIST(OUT)  -
        ON(28,6,PD) ON(18,4,ZD) ON(1,15,CH)
* Print a report for the Colorado branches
DISPLAY FROM(CODASD) LIST(RPT) -
  DATE TITLE('Colorado Branches Report') PAGE -
  HEADER('City') HEADER('Profit') HEADER('Employees') -
  ON(1,15,CH) ON(28,6,PD) ON(18,4,ZD) BLANK BETWEEN(5) -
  TOTAL('Total') AVERAGE('Average')-
  MINIMUM('Lowest') COUNT('Number of cities')
* Print a report of books for individual publishers
 DISPLAY FROM(DAPUBS) LIST(SECTIONS) -
   TITLE('BOOKS FOR INDIVIDUAL PUBLISHERS') PAGE -
   HEADER('TITLE OF BOOK')  ON(1,35,CH) -
   HEADER('PRICE OF BOOK') ON(170,4,BI,C1,F'$') -
   BTITLE('PUBLISHER:') BREAK(106,4,CH) -
   BAVERAGE('AVERAGE FOR THIS PUBLISHER') -
   BTOTAL('TOTAL FOR THIS PUBLISHER') -
   AVERAGE('AVERAGE FOR ALL PUBLISHERS') -
   TOTAL('TOTAL FOR ALL PUBLISHERS')
* Print the count of books in use from each publisher
OCCUR FROM(BKIN) LIST(PUBCT) BLANK -
  TITLE('Books from Publishers') DATE(DMY.) -
  HEADER('Publisher') HEADER('Books Used') -
  ON(106,4,CH) ON(VALCNT,N05)
* Separate output containing records for publishers
* with more than 4 books in use
SELECT FROM(BKIN) TO(BKOUT) ON(106,4,CH) HIGHER(4)
* Reformat REGION.IN1 to T1 so it can be spliced
  COPY FROM(REGNIN1) TO(T1) USING(CTL1)
* Reformat REGION.IN2 to T1 so it can be spliced
  COPY FROM(REGNIN2) TO(T1) USING(CTL2)
* Splice records in T1 with matching ON fields
  SPLICE FROM(T1) WITHALL -
    ON(5,5,CH)   -   Region
    WITH(1,4)    -   Office
    WITH(25,4)   -   Employees
    WITH(29,10)  -   Evaluation
    TO(REGNOUT)
/*
//ALL DD DSN=A123456.SORT.BRANCH,DISP=SHR
//BKS    DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//       DD DSN=A123456.SORT.SAMPADD,DISP=SHR
```

```
//DAPUBS DD DSN=&&DSRT,DISP=(,PASS),SPACE=(CYL,(2,2)),
//  UNIT=SYSDA
//PRPUBS DD SYSOUT=A
//SPUBCNTL DD *
 SORT FIELDS=(106,4,A,1,75,A),FORMAT=CH
 INCLUDE COND=(106,4,EQ,C'VALD',OR,106,4,EQ,C'WETH'),
         FORMAT=CH
/*
//CACOCNTL DD *
  SORT FIELDS=(1,15,CH,A)
  OUTFIL FNAMES=(CADASD,CATAPE),INCLUDE=(16,2,CH,EQ,C'CA')
  OUTFIL FNAMES=(CODASD,COTAPE),INCLUDE=(16,2,CH,EQ,C'CO')
/*
//CADASD DD DSN=&&CA,DISP=(,PASS),SPACE=(CYL,(2,2)),
//  UNIT=3390
//CATAPE DD DSN=CA.BRANCH,UNIT=3480,VOL=SER=111111,
//  DISP=(NEW,KEEP),LABEL=(,SL)
//CODASD DD DSN=&&CO,DISP=(,PASS),SPACE=(CYL,(2,2)),
//  UNIT=3390
//COTAPE DD DSN=CO.BRANCH,UNIT=3480,VOL=SER=222222,
//  DISP=(NEW,KEEP),LABEL=(,SL)
//OUT DD SYSOUT=A
//RPT DD SYSOUT=A
//SECTIONS DD SYSOUT=A
//BKIN DD DSN=A123456.SORT.SAMPIN,DISP=SHR
//PUBCT DD SYSOUT=A
//BKOUT DD DSN=A123456.BOOKS1,DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,(3,3)),UNIT=3390
//REGNIN1 DD DSN=A123456.REGION.IN1,DISP=SHR
//REGNIN2 DD DSN=A123456.REGION.IN2,DISP=SHR
//T1 DD DSN=&&T1,UNIT=3390,SPACE=(CYL,(5,5)),DISP=(MOD,PASS)
//REGNOUT DD DSN=A123456.REGION.OUT,DISP=(NEW,CATLG,DELETE),UNIT=3390,
//  SPACE=(TRK,(5,5))
```

ICETOOL can be called directly, as in the example above, or from a program using a parameter list to which ICETOOL can additionally return information for further processing.

What kind of reports can I produce with ICETOOL?

Professor Sort says ...

With DFSORT's ICETOOL utility, you can produce reports quickly and easily. You can produce detailed reports with section headings and statistics, page headings and statistics, and a summary page with headings and statistics. You can produce reports that identify and count unique values, duplicate values, or values that occur a specific number of times. You can also use ICETOOL to create a report showing the DFSORT installation defaults selected at your site.

Here's an example of a complete ICETOOL job. It uses the DISPLAY operator to produce a monthly revenue report:

```
//REVENUE JOB ...
//RUN EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//IN2 DD DSN=REVENUE.INPUT,DISP=SHR
//REVRPT DD SYSOUT=*
//TOOLIN DD *
* Print monthly revenue report
 DISPLAY FROM(IN2) LIST(REVRPT) DATE -
   TITLE('Monthly Revenue Report') -
   TITLE('for October') PAGE -
   HEADER('Location') ON(1,15,CH) -
   HEADER('Revenue') ON(22,6,PD,A1) -
   HEADER('Profit/Loss') ON(28,6,PD,A1) -
   TOTAL('Totals') AVERAGE('Averages') -
   COUNT('Number of Locations:') EDCOUNT(U03)
/*
```

Here's the report in the REVRPT data set that might be produced by this DISPLAY operator:

```
 10/15/08        Monthly Revenue Report        - 1 -

                      for October

 Location                        Revenue            Profit/Loss
 --------------   --------------------   --------------------
 Los Angeles                      22,530                -4,278
 San Francisco                    42,820                 6,832
 Fort Collins                     12,300                -2,863
 Sacramento                       42,726                 8,276
 ...                                 ...                   ...
 Totals                           329,637                50,665

 Averages                         27,469                 4,222

 Number of Locations:  11
```

Notice that the Revenue and Profit/Loss numbers have commas and negative signs. The numbers look like this because the "A1" editing mask was specified for the corresponding ON fields. ICETOOL has 33 different editing masks encompassing many of the numeric notations used throughout the world.

Here's another example of ICETOOL control statements. In this example, ICETOOL's OCCUR operator is used to identify userids that appear more than 4 times in a data set as possible system intruders:

```
OCCURS FROM(FAILURS) LIST(SECURITY) BLANK -
   PAGE TITLE('Possible System Intruders) DATE(DM4.) -
   HEADER('Userid') HEADER('Logon Failures') -
   ON(23,8,CH)      ON(VALCNT) -
   HIGHER(4)
```

Here's the report in the SECURITY data set that might be produced by this OCCUR operator:

```
 - 1 -   Possible System Intruders        21.09.2008

 Userid      Logon Failures
 --------   ---------------
 B7234510                 5
 D9853267                11
 ...                    ...
```

Here's an example of a DISPLAY operator that uses some additional operands that let you control the way your report looks:

```
DISPLAY FROM(ACCTS) LIST(FANCY) -
   TITLE('Accounts Report for First Quarter') -
   DATE(MD4/) BLANK -
   HEADER('Amount') ON(12,6,ZD,C1,U08) -
   HEADER('Id') ON(NUM,U02) -
   HEADER('Acct#') ON(31,3,PD,NOST,LZ) -
   HEADER('Date') ON(1,4,ZD,E'99/99',NOST) -
   INDENT(2) BETWEEN(5) -
   STATLEFT -
   TOTAL('Total for Q1') -
   AVERAGE('Average for Q1')
```

Here's the report in the FANCY data set that might be produced by this DISPLAY operator:

```
 Accounts Report for First Quarter        03/31/2008

                         Amount      Id      Acct#      Date
                      -----------     ---     ------     -----
                          932.71       1      15932     01/06
                        1,376.22       2      00187     01/28
                          831.47       3      15932     02/12
                        1,832.61       4      02158     02/17
                          763.89       5      00187     03/05
                        9,200.13       6      15932     03/19

 Total for Q1          14,937.03

 Average for Q1         2,489.50
```

INDENT, BETWEEN and STATLEFT change the default spacing for the report. Formatting items C1 (editing mask), E'99/99' (edit pattern), U08 and U02 (number of digits), NOST (no statistics) and LZ (leading zeros) change the default appearance of various numeric values.

What is ICEGENER and how can I use it?

Professor Sort says ...

ICEGENER is a DFSORT feature that couldn't be easier to use, yet provides excellent performance improvements as a replacement for IEBGENER.

Most data centers probably have hundreds of IEBGENER jobs like this one:

```
//COPYIT JOB ...
//STEP1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=FLY.INPUT,DISP=SHR
//SYSUT2 DD DSN=FLY.OUTPUT,DISP=OLD
//SYSIN DD DUMMY
```

To convert this to an ICEGENER job the "hard" way, change IEBGENER in the EXEC statement to ICEGENER. Yes, that's the hard way. The easy way is to install ICEGENER as a direct replacement for IEBGENER so that no changes to IEBGENER jobs are required! ICEGENER is then called automatically whenever IEBGENER is requested either directly or from a program.

ICEGENER uses DFSORT to process IEBGENER jobs when possible and transfers control to IEBGENER when DFSORT can't be used. Most IEBGENER jobs that use DUMMY for SYSIN can be processed by DFSORT, resulting in significant performance improvements. As an added benefit, DFSORT issues messages containing useful information such as the number of records copied and the RECFM, LRECL, and BLKSIZE of the SYSUT1 and SYSUT2 data sets.

Installing ICEGENER as a direct replacement for IEBGENER even lets RACF's IRRUT200 utility take advantage of ICEGENER's performance improvements.

In some cases IEBGENER cannot copy a SYSUT1 data set to a SYSUT2 data set with a different logical record length. ICEGENER normally copies such data sets by padding or truncating the records, issuing an accompanying warning message and return code 0. However, installation options GNPAD and GNTRUNC can be used to have ICEGENER issue a return code 4 or even transfer control to IEBGENER when the SYSUT1 and SYSUT2 logical record lengths are different.

What is OUTFIL and how can it help me?

Professor Sort says ...

OUTFIL is a very versatile DFSORT control statement. It allows you to create one or more output data sets for a sort, copy or merge application from a single pass over one or more input data sets, which can result in significant performance improvements.

OUTFIL provides a rich set of features, including a variety of reformatting, subsetting and reporting capabilities, that can be used with one output data set or many output data sets to increase programmer productivity by eliminating programming work.

What are DFSORT symbols and how can they help me?

Professor Sort says ...

A symbol is a name (preferably something meaningful) you can use to represent a field, constant or output column. Sets of symbols, also called mappings, can be used to describe a group of related fields, constants and output columns, such as the information in a particular type of record. Such mappings allow you to refer to fields, constants and output columns by their symbols, freeing you from having to know the position, length and format of a field, the value of a constant or the position of an output column you want to use.

You can even use system symbols (for example, &JOBNAME.), SET symbols and PROC symbols in your symbol constants.

DFSORT's symbol processing feature gives you a powerful, simple and flexible way to create symbol mappings for your own frequently used data. In addition, you can obtain IBM-created symbol mappings and sample jobs for data associated with RACF, DFSMSrmm and DCOLLECT. See the DFSORT website for details.

Symbols turn DFSORT's syntax into a high level language. Symbols can help to standardize your DFSORT applications and increase your productivity. Once you create or obtain symbol mappings, you can use the symbols they define anywhere you can use a field, constant or output column in any DFSORT control statement or ICETOOL operator. DFSORT symbols can be up to 50 characters, are case-sensitive and can include underscore and hyphen characters. Thus, you can create meaningful, descriptive names for your symbols, such as Price_of_Item (or Price-of-Item), making them easy to remember, use and understand.

Here's an example of the JCL and control statements for an ICETOOL job that uses symbols. A complete explanation of this example can be found in *z/OS DFSORT: Getting Started*. A complete explanation of DFSORT's Symbols feature can be found in *z/OS DFSORT Application Programming Guide*.

```
//SYM2     JOB   A492,PROGRAMMER
//TOOL     EXEC  PGM=ICETOOL,REGION=1024K
//STEPLIB  DD    DSN=A492.SM,DISP=SHR
//TOOLMSG  DD    SYSOUT=A
//DFSMSG   DD    SYSOUT=A
//SYMNAMES DD    DSN=DSN=A123456.SORT.SYMBOLS,
//               DISP=SHR
//SYMNOUT  DD    SYSOUT=*
//IN       DD    DSN=A123456.SORT.SAMPIN,
//               DISP=SHR
//OUT      DD    DSN=A123456.SORT.SAMPOUT,
//               DISP=SHR
//TOOLIN   DD    *
  SORT FROM(IN) TO(OUT) USING(CTL1)
  RANGE FROM(OUT) ON(Price) LOWER(Discount)
  RANGE FROM(OUT) ON(Price) HIGHER(Premium)
//CTL1CNTL DD *
  INCLUDE COND=(Course_Number,EQ,
                Beginning_Economics,OR,
                Course_Number,EQ,
                Advanced_Sociology)
  SORT FIELDS=(Title,A,
               Instructor_Last_Name,A,
               Instructor_Initials,A,
               Price,A)
```

How can I produce ICETOOL reports without ANSI carriage control characters?

When you produce DISPLAY or OCCUR reports, ICETOOL inserts ANSI carriage control characters in the first byte of each record that indicate actions to be taken on a printer (e.g. page eject, skip a line, etc), and also ensures that the RECFM of the report data set contains 'A' (e.g. FBA) for ANSI processing. If you want to suppress the ANSI carriage control characters, and have the RECFM set to FB instead of FBA, just specify the NOCC operand for the DISPLAY or OCCUR operator. For example:

```
DISPLAY FROM(IN) LIST(RPT) NOCC ...
```

How can I produce OUTFIL reports without ANSI carriage control characters?

Professor Sort says ...

When you produce reports with OUTFIL, DFSORT inserts ANSI carriage control characters in the first byte of each record that indicate actions to be taken on a printer (e.g. page eject, skip a line, etc), and also ensures that the RECFM of the OUTFIL data set contains 'A' (e.g. FBA) for ANSI processing. But sometimes you want to use OUTFIL's HEADERx or TRAILERx options to create data records instead of report records (e.g. initial identification records before the regular data records, or statistics records after the data records) that you can pass to another program or step. Since you aren't printing these records, you don't want the ANSI printer controls in the first byte of the output records and you want the RECFM of the output data set to be FB rather than FBA. You can get what you want by specifying the REMOVECC option.

For example, suppose you wanted to produce a RECFM=FB output data set with one record containing a count of the records in the input data set, so you could pass this record to another program. You might try using:

```
OUTFIL FNAMES=TOTCNT,NODETAIL,
 TRAILER1=(COUNT=(M11,LENGTH=6))
```

but if your input data set had 5723 records, your output record would contain:

1005723

where the '1' in the first byte is the ANSI carriage control character for a page eject and '005723' is the count of records. In addition, the RECFM would be FBA.

If you changed your OUTFIL statement to:

```
OUTFIL FNAMES=TOTCNT,NODETAIL,REMOVECC,
 TRAILER1=(COUNT=(M11,LENGTH=6))
```

DFSORT would remove the ANSI carriage control character from the first byte, so your output record would contain:

005723

and your RECFM would be FB, as required for the other program.

How can I suppress page ejects in OUTFIL reports?

Professor Sort says ...

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

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

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

For example, suppose you wanted to suppress the page eject between the report header and the first page header. You might use BLKCCH2 like this:

```
OUTFIL FNAMES=RPT2,
  HEADER1=(30:'January Report',2/),
  BLKCCH2,
  HEADER2=(5:'Account Number',25:'Name'),
  ...
```

How can I convert a VB data set to an FB data set?

Professor Sort says ...

The VTOF and BUILD operands of OUTFIL can be used to change variable-length (e.g. VB) input records to fixed-length (e.g. FB) output records. VTOF indicates that conversion is to be performed and BUILD defines the reformatted records. All output data sets for which VTOF is used must have or will be given fixed-length record formats.

Here's an example of OUTFIL conversion:

```
SORT FIELDS=(7,8,CH,A)
OUTFIL FNAMES=FB1,VTOF,BUILD=(5,76)
```

The fixed-length output records for the FB1 data set will contain positions 5-80 of the variable-length input records.

But what if some of the variable-length input records are "short" that is, they contain less than the 80 bytes specified for use by BUILD via 5,76? No problem. DFSORT automatically uses the VLFILL=C' ' option with VTOF to replace missing bytes in "short" OUTFIL BUILD fields with blanks. So all of your short OUTFIL BUILD records will be padded with blanks to 80 bytes and all of your long records will be truncated to 80 bytes.

If you want to select your own padding byte, just specify the VLFILL=byte option. For example, here's how you'd use an asterisk as the padding byte for the previous example:

```
SORT FIELDS=(7,8,CH,A)
OUTFIL FNAMES=FB1,VTOF,BUILD=(5,76),VLFILL=C'*'
```

How can I convert an FB data set to a VB data set?

Professor Sort says ...

The FTOV operand of OUTFIL can be used to change fixed-length (e.g. FB) input records to variable-length (e.g. VB) output records. If FTOV is specified without BUILD, the entire fixed-length record is used to build the variable-length record. If FTOV is specified with BUILD, the specified fields from the fixed-length record are used to build the variable-length record. The output records will consist of a 4-byte RDW followed by the fixed-length data. All output data sets for which FTOV is used must have or will be given variable-length record formats.

Here's an example of FB to VB conversion:

```
OUTFIL FNAMES=FBVB1,FTOV
OUTFIL FNAMES=FBVB2,FTOV,BUILD=(1,10,C'=',21,10)
```

The variable-length output records for the FBVB1 data set will contain a 4-byte RDW followed by the fixed-length input record.

The variable-length output records for the FBVB2 data set will contain a 4-byte RDW followed by the characters from input positions 1-10, an '=' character, and the characters from input positions 21-30.

All of the variable-length output records created with FTOV will consist of:

RDW + input fields

Since all of the input fields from the fixed-length input records are the same, all of the variable-length output records will be the same length. But if you have trailing characters such as blanks, asterisks, binary zeros, etc, you can create true variable-length output records with different lengths by using the VLTRIM=byte option of OUTFIL. VLTRIM=byte can be used with FTOV to remove trailing bytes of the specified type from the end of the variable-length output records. Here are some examples:

```
OUTFIL FNAMES=FBVB3,FTOV,VLTRIM=C'*'
OUTFIL FNAMES=FBVB4,FTOV,VLTRIM=X'40'
OUTFIL FNAMES=FBVB5,FTOV,VLTRIM=X'00'
```

FBVB3 will contain output records without trailing asterisks. FBVB4 will contain output records without trailing blanks. FBVB5 will contain output records without trailing binary zeros.

How can I put timestamps in my output records?

Professor Sort says ...

INREC, OUTREC and OUTFIL let you generate constants for the current date and time in a variety of character/zoned decimal and packed decimal formats, giving you an easy way to add timestamps to your output records. You can place these generated date and time constants anywhere in your records you like and use as many or as few of them as you need in each record.

Character/zoned decimal date/time constants: You can use DATE, DATE=(abcd), DATENS=(abc), YDDD=(abc) and YDDDNS=(ab) to generate CH/ZD constants for the current data (that is, today's date) as follows:

- **DATE** creates a constant in the form C'mm/dd/yy' which on March 15, 2004 would be C'03/15/04'.
- **DATE=(abcd)** creates a constant in the form C'adbdc' where a, b and c can be M for mm, D for dd, Y for yy or 4 for yyyy, and d is the separator character. For example, DATE=(DM4.) creates C'dd.mm.yyyy' which on March 15, 2004 would be C'15.03.2004'.
- **DATENS=(abc)** creates a constant in the form C'abc' where a, b and c can be M for mm, D for dd, Y for yy or 4 for yyyy. For example, DATE=(DMY) creates C'ddmmyy' which on March 15, 2004 would be C'150304'.
- **YDDD=(abc)** creates a constant in the form C'acb' where a and b can be D for ddd, Y for yy or 4 for yyyy, and c is the separator character. For example, YDDD=(4D-) creates C'yyyy-ddd' which on March 15, 2004 would be C'2004-075'.
- **YDDDNS=(ab)** creates a constant in the form C'ab' where a and b can be D for ddd, Y for yy or 4 for yyyy. For example, YDDD=(DY) creates C'dddyy' which on March 15, 2004 would be C'07504'.

You can use TIME, TIME=(abc) or TIMENS=(ab) to generate CH/ZD constants for the current time (that is, the time of the run) as follows:

- **TIME** creates a constant in the form C'hh:mm:ss' (24-hour time) which at 01:55:43 PM would be C'135543'.

- **TIME=(24c)** creates a constant in the form C'hhcmmcss' (24-hour time) where c is the separator character. For example, TIME=(24.) creates C'hh.mm.ss' which at 01:55:43 PM would be C'13.55.43'.
- **TIME=(12c)** creates a constant in the form C'hhcmmcss xx' (12-hour time) where c is the separator character and xx is 'am' or' pm'. For example, TIME=(12.) creates C'hh.mm.ss xx' which at 01:55:43 PM would be C'01.55.43 pm'.
- **TIMENS=(24)** creates a constant in the form C'hhmmss' (24-hour time) which at 01:55:43 PM would be C'135543'.
- **TIMENS=(12)** creates a constant in the form C'hhmmss xx' (12-hour time) where xx is 'am' or' pm' which at 01:55:43 PM would be C'015543 pm'.

You can also use DATEn and DATEn(c) to generate CH/ZD constants for the current date (that is, today's date) and TIMEn and TIMEn(c) to generate CH/ZD constants for the current time (that is, the time of the run). The separator (c) can be any character except blank. Here are tables that show the character/zoned decimal constant generated for each DATEn and DATEn(c) operand along with an example using / for c where relevent, and for each TIMEn and TIMEn(c) operand along with an example using : for c where relevent.

Operand	**Constant**	**Type**	**March 15, 2004 01:55:43 PM**
DATE1	C'yyyymmdd'	CH, ZD	C'20040315'
DATE1(c)	C'yyyycmmcdd'	CH	C'2004/03/15'
DATE2	C'yyyymm'	CH, ZD	C'200403'
DATE2(c)	C'yyyycmm'	CH	C'2004/03'
DATE3	C'yyyyddd'	CH, ZD	C'2004075'
DATE3(c)	C'yyyycddd'	CH	C'2004/075'
DATE4	C'yyyy-mm-dd-hh.mm.ss'	CH	C'2004-03-15-13.55.43'
DATE5	C'yyyy-mm-dd-hh.mm.ss.nnnnnn'	CH	C'2004-03-15-13.55.43.521386'

Operand	**Constant**	**Type**	**01:55:43 PM**
TIME1	C'hhmmss'	CH, ZD	C'135543'
TIME1(c)	C'hhcmmcss'	CH	C'13:55:43'
TIME2	C'hhmm'	CH, ZD	C'1355'
TIME2(c)	C'hhcmm'	CH	C'13:55'
TIME3	C'hh'	CH, ZD	C'13'

If you wanted to create two different timestamps at the start of each output record as follows:

```
C'TS1 IS yyyy-mm-dd hh:mm:ss, TS2 IS yyyydddhhmm '
```

you could use this OUTFIL statement:

```
OUTFIL BUILD=(C'TS1 IS ',DATE1(-),X,TIME1(:),
             C', TS2 IS ',DATE3,TIME2,X,
             1,80)
```

Packed decimal date/time constants: You can use DATEnP to generate PD constants for the current date (that is, today's date). You can use TIMEnP to generate constants for the current time (that is, the time of the run). Here are tables that show the packed decimal constant generated for each DATEnP and TIMEnP operand along with examples.

Operand	Constant	Type:	March 15, 2004
DATE1P	P'yyyymmdd'	PD	P'20040315'
DATE2P	P'yyyymm'	PD	P'200403'
DATE3P	P'yyyyddd'	PD	P'2004075'

Operand	Constant	Type	01:55:43 PM
TIME1P	P'hhmmss'	PD	P'135543'
TIME2P	P'hhmm'	PD	P'1355'
TIME3P	P'hh'	PD	P'13'

If you wanted to place a P'hhmm' constant for the time of the DFSORT run in positions 21-23 of your output records, and a P'yyyymm' constant for the date of the DFSORT run in positions 55-58 of your output records, you could use the following OUTREC statement:

```
OUTREC BUILD=(5:16,10,21:TIME2P,28:37,20,55:DATE2P,65:1,8)
```

How can I make SMF, TOD and ETOD date and time values readable?

Professor Sort says ...

DFSORT allows the use of a wide variety of character and numeric formats. DFSORT's SMF date formats (DT1, DT2 and DT3), TOD date formats (DC1, DC2 and DC3), ETOD date formats (DE1, DE2 and DE3), SMF time formats (TM1, TM2, TM3 and TM4), TOD time formats (TC1, TC2, TC3 and TC4) and ETOD time formats (TE1, TE2, TE3 and TE4) allows INREC, OUTREC, OUTFIL, and ICETOOL's DISPLAY and OCCUR operators to display the normally unreadable SMF, TOD and ETOD date and time values in a wide range of recognizable ways as follows:

Format	Result
DT1	SMF date interpreted as Z'yyyymmdd'
DT2	SMF date interpreted as Z'yyyymm'
DT3	SMF date interpreted as Z'yyyyddd'
DC1	TOD (STCK) date interpreted as Z'yyyymmdd'
DC2	TOD (STCK) date interpreted as Z'yyyymm'

Format	Result
DC3	TOD (STCK) date interpreted as Z'yyyyddd'
DE1	ETOD (STCKE) date interpreted as Z'yyyymmdd'
DE2	ETOD (STCKE) date interpreted as Z'yyyymm'
DE3	ETOD (STCKE) date interpreted as Z'yyyyddd'
TM1	SMF time interpreted as Z'hhmmss'
TM2	SMF time interpreted as Z'hhmm'
TM3	SMF time interpreted as Z'hh'
TM4	SMF time interpreted as Z'hhmmssxx'
TC1	TOD (STCK) time interpreted as Z'hhmmss'
TC2	TOD (STCK) time interpreted as Z'hhmm'
TC3	TOD (STCK) time interpreted as Z'hh'
TC4	TOD (STCK) time interpreted as Z'hhmmssxx'
TE1	ETOD (STCKE) time interpreted as Z'hhmmss'
TE2	ETOD (STCKE) time interpreted as Z'hhmm'
TE3	ETOD (STCKE) time interpreted as Z'hh'
TE4	ETOD (STCKE) time interpreted as Z'hhmmssxx'

This makes it easy to show SMF, TOD and ETOD date and time values in meaningful ways.

For ICETOOL's DISPLAY and OCCUR operators, the interpreted values can be further edited using various formatting items such as E'pattern'. This makes it easy to show SMF, TOD and ETOD date and time values in meaningful ways.

The following example shows how ICETOOL's DISPLAY operator can be used to show SMF date and time values as easily understood data in a report on SMF type-14 records.

```
DISPLAY FROM(SMF14) LIST(SMF14RPT) -
  TITLE('SMF Type-14 Records') DATE(4MD/) -
  HEADER('Date') ON(11,4,DT1,E'9999/99/99') -
  HEADER('Time') ON(7,4,TM1,E'99:99:99') -
  HEADER('Sys') ON(15,4,CH) -
  HEADER('Jobname') ON(19,8,CH) -
  HEADER('Datasetname') ON(69,44,CH)
```

The SMF14RPT report might look as follows:

```
SMF Type-14 Records        2008/02/21

      Date       Time   Sys    Jobname    Datasetname
----------   --------   ----   --------   ------------
2004/01/09   06:03:15   ID03   JOB00003   SYS1.QRS
2004/01/09   10:03:22   ID02   JOB00002   SYS1.XYZ
2004/02/10   14:05:37   ID03   JOB00004   SYS1.MNO
2004/02/21   22:11:00   ID03   JOB00005   SYS1.MNO
2004/02/21   23:12:08   ID03   JOB00006   SYS1.MNO
```

For INREC, OUTREC and OUTFIL, the interpreted values can be further edited using the pre-defined edit masks (M0-M26) or specified edit patterns you define, or converted to other formats using TO=fo.

The following example shows how the OUTREC statement can be used to convert TOD date and time values to readable form:

```
OUTREC BUILD=(X,11,8,DC3,EDIT=(TTTT-TTT),
              X,11,8,TC2,EDIT=(TT:TT))
```

The SORTOUT output might look as follows:

```
2004-009 06:03
2004-009 10:03
2004-041 14:05
2004-052 22:11
2004-052 23:12
```

How can I use INCLUDE/OMIT with numeric or non-numeric values?

Professor Sort says ...

INCLUDE and OMIT can test fields for numerics (field,EQ,NUM) or non-numerics (field,NE,NUM) of the following forms:

- Character (FS format): Test for '0'-'9' in all bytes. For example, if you used the following INCLUDE statement:

  ```
  INCLUDE COND=(21,5,FS,EQ,NUM)
  ```

  a record with a value of '02579' in positions 21-25 would be included, whereas records with values of '02A79', '-2579' and '0257 ' would not be included.

- Zoned decimal (ZD format): Test for X'F0'-X'F9' for all bytes except the last, and X'F0'-X'F9', X'D0'-X'D9' or X'C0'-X'C9' in the last (sign) byte. For example, if you used the following OMIT statement:

  ```
  OMIT COND=(31,3,ZD,NE,NUM)
  ```

  a record with a value of X'F1F3D8' in positions 31-33 would not be omitted, whereas records with values of X'F1F3A8' and X'404040' would be omitted.

- Packed decimal (PD format): Test for 0-9 for all digits and F, D or C for the sign. For example, if you used the following OUTFIL statement:

  ```
  OUTFIL INCLUDE=(11,2,PD,EQ,NUM)
  ```

  a record with a value of X'832C' in positions 11-12 would be included, whereas records with values of X'3A2F' and X'4290' would not be included.

This makes it easy to include or omit records based on whether they contain valid numeric data.

Note: You can also use NUM in the WHEN, BEGIN and END operands of IFTHEN clauses.

How can I use INCLUDE/OMIT with current, past and future dates?

Professor Sort says ...

INCLUDE and OMIT can compare appropriate fields against generated run-time constants for current, past and future dates in the following forms (d is days, m is months and c is any character except a blank):

Operand	Constant
DATE1, DATE1-d, DATE1+d	C'yyyymmdd'
DATE1(c), DATE1(c)-d, DATE1(c)+d	C'yyyycmmcdd'
DATE1P, DATE1P-d, DATE1P+d	+yyyymmdd
DATE2, DATE2-m, DATE2+m	C'yyyymm'
DATE2(c), DATE2(c)-m, DATE2(c)+m	C'yyyycmm'
DATE2P, DATE2P-m, DATE2P+m	+yyyymm
DATE3, DATE3-d, DATE3+d	C'yyyyddd'
DATE3(c), DATE3(c)-d, DATE3(c)+d	C'yyyycddd'
DATE3P, DATE3P-d, DATE3P+d	+yyyyddd
DATE4	C'yyyy-mm-dd-hh.mm.ss'
Y'DATE1', Y'DATE1'-d, Y'DATE1'+d	Y'yymmdd'
Y'DATE2', Y'DATE2'-m, Y'DATE2'+m	Y'yymm'
Y'DATE3', Y'DATE3'-d, Y'DATE3'+d	Y'yyddd'

This makes it easy to include or omit records based on whether they contain dates equal to, lower than or higher than the date of the run, a date before the run or a date after the run.

DATEn and DATEn(c) generate a character string (C'string') for today's date that can be used in comparisons just like any other character string. Likewise, DATEn-r and DATEn(c)-r generate a character string for a past date and DATEn+r and DATEn(c)+r generate a character string for a future date.

DATEnP generates a decimal number (+n) for today's date that can be used in comparisons just like any other decimal number. Likewise, DATEnP-r generates a decimal number for a past date and DATEnP+r generates a decimal number for a future date.

Y'DATEn' generates a Y constant (Y'string') for today's date that can be used in date comparisons just like any other Y constant. Likewise, Y'DATEn'-r generates a Y constant for a past date and Y'DATEn'+r generates a Y constant for a future date.

For example, if you used the following INCLUDE statement for a DFSORT run on January 29, 2006:

```
INCLUDE COND=(21,8,ZD,GE,DATE1P-7,AND,21,8,ZD,LE,DATE1P)
```

the generated date for DATE1P-7 would be +20060122 and the generated date for DATE1P would be +20060129. The SORTOUT data set would include only those records with a date in positions 21-28 between +20060122 and +2006129. So a record with a Z'20060125' date would be included, whereas a record with a Z'20060120' date would not be included.

Note: You can also use current, past and future dates in the WHEN, BEGIN and END operands of IFTHEN clauses.

How can I use INCLUDE/OMIT with "short" fields?

Professor Sort says ...

DFSORT's VLSCMP option allows short INCLUDE and OMIT fields to be compared as if they were temporarily padded with binary zeros. A short field is one where the variable-length record is too short to contain the entire field, that is, the field extends beyond the record. To illustrate what you can do with VLSCMP, consider the following INCLUDE statement:

```
INCLUDE COND=(6,1,CH,EQ,C'1',OR,21,5,CH,EQ,C'ABCDE')
```

and a 15-byte record with a character 1 in position 6.

- By default, DFSORT uses options NOVLSCMP and NOVLSHRT, which result in termination when the short INCLUDE field at positions 21-25 is found.
- If you use NOVLSCMP and VLSHRT, DFSORT treats the entire logical expression as false when the short INCLUDE field at positions 21-25 is found, so the record is omitted (this can be very useful in certain situations).
- But if you use VLSCMP, DFSORT temporarily pads the short INCLUDE field at positions 21-25 with binary zeros and does the two comparisons. The first comparison is true (position 6 has a character 1), so the record is included even though the second comparison involves a short field.

The use of binary zero padding with VLSCMP can make a comparison true instead of false, so you need to allow for that or even take advantage of it. For example, you could use statements like:

```
OPTION VLSCMP
INCLUDE COND=(6,1,CH,EQ,C'1',OR,21,1,BI,GT,X'08')
```

to include records with character 1 in position 6 or hex 09 to hex FF in position 21. Records shorter than 21 bytes without character 1 in position 6 are omitted because position 21 is padded with hex 00 and is thus less than hex 08.

On the other hand, statements like:

```
OPTION VLSCMP
INCLUDE COND=(6,1,CH,EQ,C'1',OR,21,1,BI,LT,X'08')
```

inadvertently include records with short fields, as well as records with character 1 in position 6 or hex 00 to hex 07 in position 21. Records shorter than 21 bytes are included because position 21 is padded with hex 00 and is thus less than hex 08. To omit the unwanted records with short fields, you can add a test of the record length like so:

```
OPTION VLSCMP
INCLUDE COND=(6,1,CH,EQ,C'1',OR,
              (1,2,BI,GE,X'0015',AND,21,1,BI,LT,X'08'))
```

Now records shorter than 21 bytes (hex 0015) without character 1 in position 6 are omitted, making the padding of position 21 for records with short fields irrelevent.

When you use VLSCMP, keep in mind that short fields are padded with binary zeros and construct your INCLUDE or OMIT conditions accordingly to get the results you want.

Can DFSORT use large tape block sizes?

Professor Sort says ...

DFSORT can use tape data sets with block sizes greater than 32760 bytes for input and output, providing improved performance and tape utilization.

The SDB installation and run-time option allows selection of system-determined optimum block sizes greater than 32760 bytes for output tape data sets. If you want to use system-determined block sizes for DASD and tape output data sets, specify one of the following values:

- **SDB=LARGE** if you want DFSORT to select tape output block sizes greater than 32760 bytes.
- **SDB=YES** or **SDB=SMALL** if you want DFSORT to select tape output block sizes up to 32760 bytes.
- **SDB=INPUT** if you want DFSORT to select tape output block sizes greater than 32760 bytes only if the tape input block size is greater than 32760 bytes. SDB=INPUT is the IBM-supplied installation default.

If you don't want DFSORT to use system-determined block sizes, specify **SDB=NO** (not recommended).

Even with SDB=LARGE or SDB=INPUT, DFSORT will not select a tape output block size greater than the BLKSZLIM in effect, so you may need to specify a value like BLKSZLIM=1G in your output DD statement. Here's an example of a DFSORT job that selects system-determined block sizes greater than 32760 bytes for SORTOUT and OUTFIL tape output data sets:

```
//EXAMP   JOB ...
//SDBOUT EXEC PGM=ICEMAN
//SYSOUT  DD SYSOUT=*
//SORTIN  DD  DSN=INPUT.DATA,DISP=SHR
//SORTOUT DD  DSN=OUTPUT1,DISP=(NEW,KEEP),UNIT=3590,VOL=SER=075834,
//            LABEL=(,SL),BLKSZLIM=1G
//OUT2    DD  DSN=OUTPUT2,DISP=(NEW,KEEP),UNIT=3590,VOL=SER=075835,
//            LABEL=(,SL),BLKSZLIM=1G
//SYSIN DD *
  OPTION SDB=LARGE
  SORT FIELDS=(10,8,CH,A)
  OUTFIL FNAMES=OUT1,OMIT=(22,3,CH,EQ,C'FLY')
/*
```

DFSORT's ICEGENER, like IEBGENER, will use the parameters SDB=LARGE, SDB=YES, SDB=SMALL, SDB=INPUT and SDB=NO if you specify them. Here's an example of an IEBGENER job that selects a system-determined block size greater than 32760 bytes for a SYSUT2 tape output data set:

```
//EXAMP   JOB ...
//SDBOUT EXEC PGM=IEBGENER,PARM='SDB=LARGE'
//SYSPRINT DD SYSOUT=*
//SYSUT1  DD  DSN=INPUT.DATA,DISP=SHR
//SYSUT2  DD  DSN=OUTPUT3,DISP=(NEW,KEEP),UNIT=3590,VOL=SER=075836,
//            LABEL=(,SL),BLKSZLIM=1G
//SYSIN DD DUMMY
```

This same job can use DFSORT's more efficient ICEGENER facility if your site has installed ICEGENER to be invoked by the name IEBGENER. Alternatively, you can specify PGM=ICEGENER to ensure that ICEGENER is used.

How can I recover data sets with incomplete spanned records?

Professor Sort says ...

DFSORT's SPANINC installation and run-time option lets you decide how to handle data sets that contain incomplete spanned records. Here are the available options:

- SPANINC=RC0 - specifies that DFSORT should issue a warning message, set a return code of 0 and eliminate all incomplete spanned records it detects. Valid records (that is, complete spanned records) are recovered and written to the output data set.
- SPANINC=RC4 - specifies that DFSORT should issue a warning message, set a return code of 4 and eliminate all incomplete spanned records it detects. Valid records (that is, complete spanned records) are recovered and written to the output data set.
- SPANINC=RC16 - specifies that DFSORT should issue an error message, set a return code of 16 and terminate if it detects an incomplete spanned record.

How many work and merge data sets can DFSORT use?

Professor Sort says ...

DFSORT allows you to use up to 255 dynamically allocated work data sets. Any valid ddname of the form SORTWKdd or SORTWKd can be used for work data sets. In addition to the standard ddnames like SORTWK00-99 and SORTWK0-9, you can also use ddnames like SORTWK3B, SORTWK#5, SORTWKXY and SORTWKZ.

DFSORT allows you to merge up to 100 input data sets.

DFSORT also supports large format data sets which were introduced with z/OS 1.7. Prior to z/OS 1.7, most sequential data sets were limited to 65,535 tracks on each volume, although most hardware storage devices supported far more tracks per volume. To support this hardware capability, z/OS 1.7 allowed users to create new large format data sets, which are physical sequential data sets with the ability to grow beyond the previous size limit. DFSORT supports large format data sets for input, output and work files. DFSORT's dynamic allocation feature automaticaly allocates work data sets as large format on z/OS 1.7 and above. To exploit large format data sets for JCL work data sets, DSNTYPE=LARGE should be added to the SORTWKdd DD statements.

The large number of supported data sets allows you to sort or merge a significant amount of data in a single application.

How much main storage does DFSORT need to sort most efficiently?

Professor Sort says ...

It depends on several factors, the most important of which is the **input file size** (that is, the amount of data being sorted). In general, the larger the file size, the more main storage DFSORT requires to sort with the same degree of efficiency. Fortunately, DFSORT's Dynamic Storage Adjustment (DSA) feature takes most of the guesswork out of setting main (virtual) storage for DFSORT.

With DSA, DFSORT uses the default (TMAXLIM) amount of main storage for most sorts, unless it determines that DFSORT performance would benefit from using more main storage. In those cases, DFSORT **automatically** uses more main storage to achieve optimum performance.

See "What is Dynamic Storage Adjustment and how can it help me?" in this paper for more information on DFSORT's DSA feature.

What is Dynamic Storage Adjustment and how can it help me?

Professor Sort says ...

Dynamic Storage Adjustment (DSA) is a feature that allows DFSORT to **automatically** use more main storage to achieve optimum performance.

The **DSA=n** installation option specifies the maximum amount of storage (in megabytes) available to DFSORT for dynamic storage adjustment of a sort application when SIZE/MAINSIZE=MAX is in effect. If you specify a DSA value greater than the TMAXLIM value, you allow DFSORT to use more storage than the TMAXLIM value if doing so should improve performance. The amount of storage DFSORT uses is subject to the DSA value as well as system limits such as region size. However, whereas DFSORT always tries to obtain as much storage as it can up to the TMAXLIM value, DFSORT only tries to obtain as much storage as needed to improve performance up to the DSA value.

The performance improvement from dynamic storage adjustment usually provides a good tradeoff against the increased storage used by DFSORT. If storage is not constrained on your system, DSA can be set to a large value (such as 64 or 128 megabytes) to optimize DFSORT performance on very large sorts. On storage constrained systems, however, the DSA value should be set low enough to prevent unacceptable paging.

The IBM shipped default for DSA is 64 megabytes.

Is it important to install the DFSORT SVC?

Professor Sort says ...

The DFSORT-supplied SVC enables DFSORT to run authorized functions without itself being authorized.

For best performance, you should install the DFSORT SVC, because the following performance-related functions are impaired if DFSORT's SVC is not available:

- SMF TYPE-16 record

  DFSORT's type-16 SMF record contains useful information for analyzing the performance of DFSORT. Without the SVC, DFSORT cannot write the SMF record to an SMF system data set. If DFSORT's SMF feature is activated (installation or run-time option SMF=SHORT or SMF=FULL) and a properly installed SVC is not available, then all DFSORT applications will abend.

- CACHE FAST WRITE

  Cache fast write (CFW) enables DFSORT to save elapsed time because DFSORT is able to write its intermediate data into storage control cache, and read it from the cache. Without the SVC, DFSORT cannot use CFW, and elapsed time performance can be degraded.

- CACHING MODE

For cached storage control units, DFSORT selects the caching mode that is best for the circumstances. This also helps other applications make the best use of the cache. Without the SVC, DFSORT cannot set these caching modes, which can result in system and DFSORT elapsed time performance degradation.

In addition to the functions described above, there are other performance enhancements that are available to DFSORT through use of the SVC.

Thus you should ensure that the installation of the DFSORT SVC has been completed correctly so that the SVC can be used. How the SVC is installed depends on whether you are replacing an earlier DFSORT release, installing a new release to coexist with an earlier release, or installing DFSORT for the first time. You must set the SVC installation option to correspond to the way you install the SVC. See *z/OS DFSORT Installation and Customization* for details on installing the DFSORT SVC.

How can DFSORT be used to analyze data produced by DCOLLECT, DFSMSrmm, etc?

Professor Sort says ...

The DFSORT product tape contains a set of illustrative examples that show how DFSORT and ICETOOL can be used to analyze data created by DFHSM, DFSMSrmm, DCOLLECT and SMF. The source for the following examples are available in sample job ICESTGEX:

- DCOLEX1 - DCOLLECT Example 1: VSAM report.
- DCOLEX2 - DCOLLECT Example 2: Conversion reports.
- DCOLEX3 - DCOLLECT Example 3: Capacity planning analysis and reports.
- DFHSMEX1 - DFHSM Example 1: Deciphering Activity Logs.
- DFHSMEX2 - DFHSM Example 2: Recover a DFHSM CDS with a broken index.
- RMMEX1 - DFSMSrmm Example 1: SMF audit report.
- RMMEX2 - DFSMSrmm Example 2: Create ADDVOLUME commands.

You can obtain the ICESTGEX examples from the DFSORT product tape, or download a .zip file containing the examples from the DFSORT FTP site using:

ftp://ftp.software.ibm.com/storage/dfsort/mvs/icestgex.zip

DFSORT symbols can be used to make it easier to process data associated with other products.

SHARE 86 presentation 3054 included the following examples of using DFSORT's ICETOOL utility to create reports from DCOLLECT data:

- SHAR861 - Example using ICETOOL to extract SMS statistics for volumes, and produce a summary report.
- SHAR862 - Example to show allocated space, used space, and unblockable space by data set by DSORG.

You can download a .zip file containing the SHR3054 examples from the DFSORT FTP site using:

ftp://ftp.software.ibm.com/storage/dfsort/mvs/shr3054.zip

Here's the DCOLEX2 example from ICESTGEX to show you a small sample of what DFSORT/ICETOOL can do with DCOLLECT data:

```
//DCOLEX2  JOB ...
//*****************************************************
//* DCOLLECT EXAMPLE 2: CONVERSION REPORTS
//*  ASSIGN ENGLISH DESCRIPTIONS TO BIT FLAGS TO SHOW
//*  MIGRATION STATUS OF VOLUMES (IN CONVERSION, MANAGED
//*  BY SMS AND NON-SMS MANAGED).
//*****************************************************
//STEP1  EXEC  PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*   ICETOOL MESSAGES
//DFSMSG   DD SYSOUT=*   DFSORT  MESSAGES
//TOOLIN   DD *          CONTROL STATEMENTS
* PART 1 - ADD IDENTIFYING STATUS (FLAG) DESCRIPTION FOR
*          'MANAGED', 'IN CONVERSION' AND 'NON-MANAGED'
*          VOLUME RECORDS
   COPY FROM(DCOLALL) USING(FLAG)
* PART 2 - PRINT REPORT SHOWING COUNT OF EACH STATUS TYPE
   OCCUR FROM(STATUS) LIST(REPORTS) -
    TITLE('STATUS COUNT REPORT') DATE -
    BLANK -
    HEADER('STATUS') ON(7,20,CH) -
    HEADER('NUMBER OF VOLUMES') ON(VALCNT)
```

```
* PART 3 - PRINT REPORT SORTED BY VOLUMES AND STATUS
  SORT FROM(STATUS) TO(SRTVOUT) USING(SRTV)
  DISPLAY FROM(SRTVOUT) LIST(REPORTV) -
    TITLE('VOLUME/STATUS REPORT') DATE PAGE -
    BLANK -
    HEADER('VOLUME') ON(1,6,CH) -
    HEADER('STATUS') ON(7,20,CH)
* PART 4 - PRINT REPORT SORTED BY STATUS AND VOLUMES
  SORT FROM(STATUS) TO(SRTIOUT) USING(SRTI)
  DISPLAY FROM(SRTIOUT) LIST(REPORTI) -
    TITLE('STATUS/VOLUME REPORT') DATE PAGE -
    BLANK -
    HEADER('STATUS') ON(7,20,CH) -
    HEADER('VOLUME') ON(1,6,CH)
//DCOLALL DD DSN=Y176398.R12.DCOLLECT,DISP=SHR
//STATUS   DD DSN=&&TEMP1,DISP=(,PASS),UNIT=SYSDA,
//  LRECL=50,RECFM=FB,DSORG=PS
//SRTVOUT  DD DSN=&&TEMP2,DISP=(,PASS),UNIT=SYSDA
//SRTIOUT  DD DSN=&&TEMP3,DISP=(,PASS),UNIT=SYSDA
//FLAGCNTL DD *
* FIND V-TYPE RECORDS WITH STATUS FLAGS OF INTEREST
  INCLUDE COND=(9,2,CH,EQ,C'V ',AND,
                35,1,BI,NE,B'......10')
* CREATE RECFM=FB OUTPUT RECORDS WITH VOLSER AND
* STATUS DESCRIPTION.
* LOOKUP/CHANGE TABLE -
*   FLAG          DESCRIPTION
*   ----------    --------------------
*   DCVMANGD      MANAGED BY SMS
*   DCVINITL      IN CONVERSION TO SMS
*   DCVNMNGD      NON-SMS MANAGED

  OUTFIL FNAMES=STATUS,CONVERT,
    OUTREC=(29,6,
      35,1,CHANGE=(20,
            B'......11',C'MANAGED BY SMS',
            B'......01',C'IN CONVERSION TO SMS',
            B'......00',C'NON-SMS MANAGED'),
      50:X)
//REPORTS DD SYSOUT=*
//SRTVCNTL DD *
* SORT BY VOLUME AND INDENTIFYING STATUS STRING
  SORT FIELDS=(1,6,CH,A,7,20,CH,A)
//SRTICNTL DD *
* SORT BY INDENTIFYING STATUS STRING AND VOLUME
  SORT FIELDS=(7,20,CH,A,1,6,CH,A)
//REPORTV DD SYSOUT=*
//REPORTI DD SYSOUT=*
//*
```

How can I create a report with just statistics?

Professor Sort says ...

Both OUTFIL and ICETOOL's DISPLAY operator can be used to produce reports with headers, columns of data, and trailers containing **overall** statistics (that is, totals, averages, minimums, maximums and/or counts for all of the data).

In addition, the NODETAIL operand of OUTFIL can be used to eliminate the columns of data so that only the headers and trailers (with overall statistics) are printed.

Here's an example of the control statements for an OUTFIL report containing **overall** statistics without the columns of data:

```
OPTION COPY
OUTFIL FNAMES=RPT,NODETAIL,
  TRAILER1=(2/,
   'Summary Report for Division Revenues',5X,DATE,3/,
   'Number of divisions reporting: ',COUNT=(M10,LENGTH=3),2/,
   'Total revenue: ',18:TOTAL=(25,8,ZD,M14,LENGTH=12),2/,
   'Lowest revenue: ',18:MIN=(25,8,ZD,M14,LENGTH=12),2/,
   'Highest revenue: ',18:MAX=(25,8,ZD,M14,LENGTH=12),2/,
   'Average revenue: ',18:AVG=(25,8,ZD,M14,LENGTH=12))
```

Here's an example of the output that might appear in RPT:

```
Summary Report for Division Revenues     10/17/08


Number of divisions reporting:  15
Total revenue:   166 679 754
Lowest revenue:   (5 213 641)
Highest revenue:  82 348 343
Average revenue:  33 335 950
```

Both OUTFIL and ICETOOL's DISPLAY operator can also be used to produce reports with headers, columns of data, and trailers containing **section** statistics (that is, totals, averages, minimums, maximums and/or counts for records with the same value for a specified field).

Again, the NODETAIL operand of OUTFIL can be used to eliminate the columns of data so that only the headers and trailers (with section statistics) are printed.

Here's an example of the control statements for an OUTFIL report containing **section** totals without the columns of data:

```
SORT FIELDS=(3,5,CH,A)
OUTFIL FNAMES=SUMMARY,NODETAIL,
 HEADER2=(' Date: ',DATE=(DMY.),4X,'Page: ',PAGE,/,
          ' Division Profit/Loss Report',2/,
          5:'Division',15:' Profit or Loss',32:'Employees',/,
          5:'--------',15:'---------------',32:'---------'),
 SECTIONS=(3,5,
  TRAILER3=(5:3,5,
           15:TOTAL=(25,7,ZD,M12,LENGTH=15),
           32:TOTAL=(15,5,ZD,M12,LENGTH=9)))
```

Here's an example of the output that might appear in SUMMARY:

```
Date: 17.10.08    Page:      1
Division Profit/Loss Report

   Division   Profit or Loss  Employees
   --------  ---------------  ---------
   East             4,675,646      1,477
   North           14,415,345      3,802
   South             -221,882        838
   West             5,996,480      1,650
```

How can I have DFSMS place my work data sets on volumes with adequate space?

Professor Sort says ...

It is common for ACS routines to check only the &size variable when determining the volume placement for a particular data set. This practice may cause each dynamically allocated SORTWK data set to register with a &size value of 0, so that they all end up on the same volume, resulting in B37 abends. The best way to ensure that the dynamically allocated SORTWKdd data sets are placed on volumes with adequate space is to modify the ACS routines to base the volume assignment decisions on the **&maxsize** variable, rather then the &size variable.

What kind of performance improvements are possible with OUTFIL?

Professor Sort says ...

With OUTFIL, sort, merge and copy applications can create multiple output data sets containing unedited or edited records, different ranges or subsets of records, reports and so on, from a single pass over one or more input data sets.

To show how the use of OUTFIL can slash elapsed time, EXCPs and CPU time, we did laboratory measurements in a stand-alone environment for the creation of 10 and 15 sorted subset output data sets of 30 megabytes from an input data set of 200 megabytes. Here's the range of performance improvements we observed with DFSORT using a single sort step with OUTFIL vs DFSORT using a sort step followed by multiple copy steps (keep in mind that performance improvements will vary depending on factors such as key size, record type, number of records, processor model, region size, and so on):

- 85% to 89% elapsed time reduction
- 13% to 30% CPU time reduction
- 89% to 92% EXCP count reduction

How can DFSORT help with the Year 2000 challenge?

Professor Sort says ...

The widespread use of two-digits to represent years can result in significant data processing problems. For example, the normal ordering of 00 before 99 is incorrect when 00 represents 2000 and 99 represents 1999. DFSORT's Year 2000 features provide tools that can help you correctly process a wide variety of dates for this century and the coming ones.

DFSORT provides powerful Year 2000 features that allow you to sort, merge, compare and transform character, zoned decimal and packed decimal dates with two-digit years according to a specified sliding or fixed century window.

You can handle two-digit year date fields, and their special indicators like zeros and nines, in the following ways:

- Set the appropriate **century window** for your applications and use it to interpret the years (yy) correctly when you sort, merge, compare or transform two-digit year dates.

  For example, set a century window of 1915-2014 or 1950-2049.

- **Order** character, zoned decimal or packed decimal two-digit year dates with the SORT and MERGE statements. For example, order 980622 (representing June 6th, 1998) before 000622 (representing June 6th, 2000) in ascending sequence, or order 000622 before 980622 in descending sequence.

  DFSORT's **full date formats** (Y2T, Y2U, Y2V, Y2W, Y2X and Y2Y) make it easy to sort or merge any type of yyx...x or x...xyy date (for example, yyq, yymm, yyddd, yymmdd, qyy, mmyy, dddyy or mmddyy). DFSORT's **year formats** (Y2C, Y2Z, Y2S, Y2P, Y2D and Y2B) are available if you need to sort or merge special dates (for example, ddmmyy) or year fields (yy).

- **Select** records by **comparing** character, zoned decimal or packed decimal two-digit year date fields to date constants or other date fields with the INCLUDE and OMIT statements or OUTFIL's INCLUDE and OMIT operands. For example, select records with a date field between January 1st, 1998 and December 31st, 2003.

  Note: You can also use two-digit year dates in the WHEN, BEGIN and END operands of IFTHEN clauses.

  DFSORT's full date formats make it easy to compare any type of yyx...x or x...xyy date field to a date constant or another date field. DFSORT's year formats are avalable if you need to compare years.

- **Transform** character, zoned decimal or packed decimal two-digit year dates to character four-digit year dates with or without separators, or transform packed decimal two-digit year dates to packed decimal four-digit year dates, with the INREC, OUTREC or OUTFIL statements. For example, transform P'99015' to C'1999015', C'1999/015' or P'1999015'.

  DFSORT's full date formats make it easy to transform any type of yyx...x or x...xyy date. DFSORT's year formats are available if you need to transform special dates or year fields.

In addition, you can use DFSORT's Year 2000 features or explicitly without MLE.

These DFSORT enhancements allow you to continue to use two-digit year dates for sorting, merging and comparing, and help you to change from using two-digit year dates to using four-digit year dates as appropriate.

'KP' WOULD EXCEED MAXIMUM SIZE.
'SORTASKP' LINE 1255: //DAPUBS DD DSN=&&DSRT,DISP=(,PASS),SPACE=(CYL,(2,2)),
STARTING PASS 2 OF 2.
'KP' WOULD EXCEED MAXIMUM SIZE.
'SORTASKP' LINE 1255: //DAPUBS DD DSN=&&DSRT,DISP=(,PASS),SPACE=(CYL,(2,2)),