Based on zOS V2.1

DFSORT is IBM's high-performance sort, merge, copy, analysis, and reporting product for z/OS.

Any or all of the defaults can be changed at any time after DFSORT is installed either with ICEPRMxx members of concatenated PARMLIB or with the ICEMAC macro. You might want to first install DFSORT with the given defaults, then tailor them to your requirements after you have run DFSORT for a while.

DFSORT allows you to maintain separate sets of installation defaults for eight different environments.

• You can write and install an initialization exit, **ICEIEXIT**, to provide more control over certain installation- and run-time options.
- Your ICEIEXIT can analyze the original installation- and run-time options together with information from system control blocks to determine whether to change certain options at run time.
For example, by using ICEIEXIT, your site can:
> Enforce an absolute limit on maximum storage for DFSORT jobs
> Increase or decrease the maximum storage for DFSORT jobs based on performance requirements or job name
> Cause DFSORT to reserve more storage for jobs with user exits that process a large number of data sets
> Use different options for test and production modes.
- When installed and activated, ICEIEXIT receives control in the initialization phase of DFSORT after all scanning and cross-checking of options is complete. DFSORTpasses a set of installation-time options and a set of run-time options to ICEIEXIT.

Note: Although an ICEIEXIT can change certain options based on the time of day, the time-of-day installation default environments provide an easier and more comprehensive method for time-of-day option controls.

What run-time options can ICEIEXIT change?

• Run-time options are those in effect for the application. Some options are specified when invoking a job; DFSORT chooses other run-time options based on the default settings established at installation time.
- Your ICEIEXIT can change certain run-time options, but keep in mind that DFSORT might override them because of conflicts between options, technique, or function restrictions, or for performance considerations.
- Several run-time options can be changed:
> Maximum storage to be used
> Storage to be reserved
> Action for critical error
> Sequence checking of output records
> Use of Hiperspace for Hipersorting
> Use of data space for dataspace sorting
> Use of a memory object for memory object sorting
> Maximum OUTFIL data set buffer space

ICEIEXIT can examine:
> Whether the application is a sort, merge, or copy
> Whether DFSORT was invoked directly or through a program
> Whether the Blockset technique is being used
> Whether storage was allocated above 16 MB virtual
> The options SIZE/MAINSIZE, RESALL/RESINV, ABEND/RC16, VERIFY/NOVERIFY, HIPRMAX, DSPSIZE, ODMAXBF, and MOSIZE.

Note: Any ICEIEXIT changes to run-time options override all changes made to these options by installation options, other run-time options, or an EFS program.

Be generous with main storage

• By default, DFSORT can use 6 megabytes of main memory for sort, merge and copy applications, or even more when appropriate.
- If your site does not have an installation default of SIZE=MAX, we recommend setting the MAINSIZE=MAX parameter for your job. As an example, you can set MAINSIZE=MAX like this:

```
//DFSPARM DD *
 OPTION MAINSIZE=MAX
/*
```

Using high-speed disks, such as IBM's Enterprise Storage Server subsystems, offers the best performance for work data sets.

• Alternatively, you can specify the MAINSIZE=*n*M parameter (*n* megabytes) to give DFSORT more or less storage for your job, but we do not recommend this unless you have a specific reason for using MAINSIZE=nM rather than MAINSIZE=MAX.

As an example, you can set MAINSIZE=2M like this:

```
//DFSPARM DD *
 OPTION MAINSIZE=2M
/*
```

With COBOL, you can enhance DFSORT performance by using the FASTSRT compile-time option. With FASTSRT, DFSORT does the input and output processing, rather than COBOL.

• By default, DFSORT can use memory objects, hiperspaces or data spaces, when appropriate, instead of or along with disk work data sets, to improve the performance of sort applications.
- If your site has an installation default of MOSIZE=0, HIPRMAX=0 or DSPSIZE=0, you might want to specify the MOSIZE=MAX, HIPRMAX=OPTIMAL and DSPSIZE=MAX parameters when you sort large data sets.
As an example, you can set these parameters like this:

```
//DFSPARM DD *
OPTION MOSIZE=MAX,HIPRMAX=OPTIMAL,DSPSIZE=MAX
/*
```

ICETOOL is a multipurpose DFSORT utility that uses the capabilities of DFSORT to perform multiple operations on one or more data sets in a single step.

The performance of a DFSORT application is largely determined by the use of a special set of product features.

- Blockset technique‡
- OUTFIL
- Memory object sorting, Hipersorting and dataspace sorting
- Dynamic Storage Adjustment
- Cache fast write
- ICEGENER
- Compression
- Striping
- Dynamic allocation of work data sets
- System-determined block size
- Larger tape blocksizes (greater than 32K)
- Managed tape data sets
- IDCAMS BLDINDEX
- DFSORT's Performance Booster for The SAS System

NOTE: The way in which COBOL interfaces with DFSORT depends on the use of COBOL features such as FASTSRT, NOFASTSRT, USING, GIVING and INPUT and OUTPUT PROCEDUREs, and DFSORT features such as COBOL exits and DFSORT control statements. In general, the features you use are dictated by the needs of your application. But in many cases, an application can achieve its results using one or another of these features, that is, you have a choice.

‡ **Efficient blocking -** You can improve the performance of DFSORT significantly by blocking your input and output records efficiently such as using system-determined optimum block sizes for your data sets.

You will get the best performance from DFSORT if you follow these guidelines:
Be *generous* with main storage.
- Allow memory object sorting, hipersorting and dataspace sorting.
- Use high-speed disks for work space.
- Eliminate unnecessary fields with INREC.
- Eliminate unnecessary records with INCLUDE or OMIT.
- Eliminate unnecessary records with STOPAFT and SKIPREC.
- Consolidate records with SUM.
- Create multiple output data sets with OUTFIL
- Replace program logic with DFSORT control statements.
- Use FASTSRT with COBOL.
- Avoid options that degrade performance.

The following parameters and options might *adversely* affect DFSORT performance. Use them only when necessary.
- VERIFY parameter
- EQUALS parameter
- NOBLKSET parameter
- CKPT parameter
- EQUCOUNT parameter
- NOCINV parameter
- LOCALE parameter
- BSAM parameter
- NOASSIST parameter
- NOCFW parameter
- Tape work data sets
- User exit routines
- EFS program
- Dynamic link-edit of user exit routines
- Small values for the MOSIZE, HIPRMAX, DSPSIZE, or MAINSIZE parameters

DFSORT offers a rich set of fast, efficient productivity features. These features can eliminate the up-front costs of writing and debugging your code to perform various tasks, & will perform those tasks efficiently.

Improving the performance of DFSORT consists of a number of activities, including:
- Tuning on a site-wide or system level
- Tuning of individual applications
- Designing efficient applications

Using initialization and termination exits

• You can use user-written, installation-wide initialization and termination exits (ICEIEXIT and ICETEXIT) to perform a variety of functions, such as overriding the options currently in effect and collecting statistical data. (The installation-wide initialization exit is subsequently referred to as an initialization exit or ICEIEXIT throughout this issue; the installation-wide termination exit is referred to as a termination exit or ICETEXIT.)

- Initialization exits - You can use sample jobs ICEIXREC and ICEIXAPP in the SICESAMP library to install an ICEIEXIT using SMP/E. **Note:** The sample SMP/E usermod in ICEIXREC will place your ICEIEXIT in the SORTLPA library.

- Termination exits - You can use sample jobs ICEIXREC and ICEIXAPP in the SICESAMP library to install an ICETEXIT using SMP/E where the sample SMP/E usermod in ICEIXREC will place your ICETEXIT in the SORTLPA library.
NOTE: Two examples of ICETEXITs are available.

DFSORT's collating behavior can be modified according to your cultural environment. Your cultural environment is defined to DFSORT using the X/Open* locale model. A locale is a collection of data grouped into categories that describe the information about your cultural environment. The collate category of a locale is a collection of sequence declarations that defines the relative order between collating elements (single character and multi-character collating elements). The sequence declarations define the collating rules.

Running DFSORT resident - By running DFSORT resident, that is, with DFSORT's SORTLPA library in LPALST and DFSORT's SICELINK library in LINKLST, you can gain three performance benefits:
• Two or more applications can use the same copy of DFSORT in main storage at the same time.
- This enables central storage to be used more efficiently and cuts down on system paging.
• The DFSORT load modules do not have to be loaded each time DFSORT is run.
- This also saves unnecessary paging and time. This is especially noticeable for the smaller DFSORT applications, which tend to make up the bulk of DFSORT jobs at most sites.
• The space for the DFSORT load modules is not charged against the virtual storage limits of individual applications.
- This saves storage that can be used by DFSORT to do a more efficient sort.

Making the DFSORT SVC available enables DFSORT to run authorized functions without itself being authorized.
- In particular, the following performance-related functions are impaired if DFSORT's SVC is not available:

SMF type-16 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, although the record can still be obtained through an ICETEXIT routine. 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 (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 issues message ICE191I, although processing continues with possibly degraded elapsed time performance.

Caching mode - For storage control units that support cache, DFSORT selects the caching mode that appears to be the best for the circumstances. Without the SVC, DFSORT cannot set these caching modes, and issues message ICE191I.
- This results in the default modes being selected, with possibly degraded system and DFSORT elapsed time performance.

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

Recommendation: Make the DFSORT SVC available for best performance.

ICEGENER - DFSORT's ICEGENER facility allows qualifying IEBGENER jobs to be routed to the more efficient DFSORT copy function.
- In most cases, using the DFSORT copy function instead of IEBGENER requires less CPU time, less elapsed time, and results in fewer EXCPs, although there are circumstances where ICEGENER will not be invoked.
Examples - A SYSIN DD statement other than SYSIN DD DUMMY is present; detection of an error before DFSORT has started the copy operation; a condition listed in DFSORT message ICE160A.

Listing the installation defaults with ICETOOL

You can use an ICETOOL job similar to the one here to list the merged PARMLIB/ICEMAC installation defaults actually in use at your site for the installation environment(s).

```
//DFRUN JOB A402,PROGRAMMER
//LISTDEF EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=A
//DFSMSG DD SYSOUT=A
//SHOWDEF DD SYSOUT=A
//TOOLIN DD *
  DEFAULTS LIST(SHOWDEF)
/*
```

Plan ahead when designing new applications

You should consider several factors when designing new applications.

Some factors are highlighted here. Whenever possible:
• Use either EBCDIC character or binary control fields
• Place binary control fields so they start and end on byte boundaries
• Avoid using the alternative collating sequence character translation
• If you know that a fixed-point control field always contains positive values, specify it as a binary field.
• If you know that a packed decimal or zoned decimal control field always contains positive values with the same sign (for example, X'C'), specify it as a binary field.
• Use packed decimal format rather than zoned decimal
• If several contiguous character or binary control fields in the correct order of significance are to be sorted or merged in the same order (ascending or descending), specify them as one control field
• Avoid overlapping control fields.
• Avoid using locale processing if your SORT, MERGE, INCLUDE, or OMIT character fields can be processed using the binary encoding of the data.

Certain options can adversely affect performance, and should be used only when necessary.

Extended format data sets have features that can:
• significantly reduce the elapsed time DFSORT spends reading and writing data
• assist in managing the space requirements of very large data sets.