A

COBOL-IT® Compiler Suite Enterprise Edition

CitSORT® Reference Manual
Version 2.5

Contents

Acknowledgment

This documentation is derived from COBOL-IT Source code, parts of which are derived from OpenCOBOL.

In 2008, COBOL-IT forked its own compiler branch, with the intention of developing a fully featured product and offering professional support to the COBOL user industry.

Conventions used in the General Format diagrams:

Brackets [ ] identify syntax elements that are supported but not required.

Curly Braces { } identify alternative syntax elements. Among syntax elements described within stacked curly braces, only one of the entries may be selected.

Ellipses (...) indicate (optional) repetition. If the syntax element is a required element, then it will be surrounded by curly braces.

Third-Party software components embedded in the SOFTWARE and Services and submitted to specific licenses:

VBISAM

Contact Information:

The Lawn
22-30 Old Bath Road
Newbury, Berkshire, RG14 1QN
United Kingdom
Tel: +44-0-1635-565-200

1 CitSORT®

1.1 Overview

In the document below, we will examine the different functions supported by CitSORT, and include examples, intended to facilitate your adoption of, or transition to CitSORT.

CitSORT is a command-line driven data transformation utility. It provides high-performance Sort alternative and powerful data transformation capabilities.

Functionally, CitSORT provides syntax for making use of multiple processors, if available, naming input files, naming rules for interpreting the input files, which can include rules for transformation, and then naming output files, for which different formats can be described.

To better understand, it helps to understand logically how the utility parses a typical command. In the simplest case, CitSORT is used to sort a file on a key represented by the "sort fields" parameter, into an output file. In the example below, we begin with a sequential file that lists the Presidents and Vice Presidents of the United States from first to most recent. Our simple sort will reverse the order of the list, and generate the output in line sequential format.

1.1.1 Setting the Temporary Directory

CitSORT uses the system's temporary directory to store the temporary files that are required for the intermediate stages of a SORT.

As a general rule, Windows, Linux and UNIX operating systems have default ways of handling temporary files in a default temporary directory.

On Windows:

TMP, or TEMP is used to specify the directory to be used for temporary files. If neither TMP or TEMP is defined, or if it is set to the name of a directory that does not exist, temporary files are created in the current working directory.

A typical path is %USERPROFILE%\AppData\Local\Temp.

There is no limitation on the size of the temporary folder in Windows. You are only limited by the overall amount of free disk space that you have.

To change the temporary directory:
SET TMP=C:\NEWPATH\TMP

On Linux/UNIX

TMPDIR is used to specify the directory to be used for temporary files. A typical path is /tmp or /var/tmp.

If you need to change the maximum size of a file, or extend a limitation in place on the size of the temporary folder in Linux/UNIX operating environment, consult with your System Administrator.

To change the temporary directory:

export TMPDIR=/usr/newpath/tmp

1.1.2 Command-line examples, Windows & Linux/Unix Considerations

The command-line examples in the Reference Manual are executed on a Windows platform where the shell (CMD.exe) is not interpreting the parentheses.

Executing this command-line in a Linux/UNIX environment, where the shell interprets parentheses, you could see an error, such as:

```
-bash: syntax error near unexpected token `('
```

This is a shell error, not a citsort error. To correct this, you can escape the parentheses with \ .

See the difference below:

In a Windows environment:

>citsort use presidents.dat record F 85 sort fields (1,2,nu,d) give pres2.txt org ls

In a Linux/Unix environment

>citsort use presidents.dat record F 85 sort fields \(1,2,nu,d\) give pres2.txt org ls

In a Linux/UNIX environment, you can avoid this error by transferring the commands into a parameter-file, and using the syntax : >citsort take parameter-file.txt, as described below.

1.1.3 Transferring commands into a parameter file

Transferring commands into a parameter-file is recommended when running citsort in Linux and UNIX environments.
Transferring command line into a parameter-file can also be useful, if your command lines are very long, and if you wish to add comments to the different parts of the command. Comments can be inserted into a parameter-file after the "*" character.

>citsort take president-params.txt (where president-params.txt contains the following:)

Use presidents.dat	*** input file to be sorted.**
Record F 85	*** fixed length, 85 bytes, sequential**
Sort Fields (1,2,nu,d)	*** sort key bytes 1-2/numeric,descending**
Give pres2.txt org ls	*** output to pres2.txt, line sequential**

1.1.4 The CitSORT Flow of Control for a SORT

As noted above, the general flow of logic is:

1. Allow for multiprocessing	-processmax, -processrec clauses
2. Name the input file(s)	USE Statement
3. Re-format the input file(s)	INREC Statement
4. Name the Sort Key(s)	SORT Statement
5. Apply Sum algorithm(s)	SUM Statement
6. Define Sort Filter(s)	INCLUDE/OMIT Statements

7. Name the output file	GIVE Statement
8. Allow different output files	OUTFIL Statement
9. Re-format output file(s)	OUTREC Statement

1.2 CitSORT® General Format

- Format 1

```
citsort
[-test]

[-processmax = int ]
[-processrec = int ]

[SORT | SORT-EBCDIC | MERGE |  COPY] [SIGN-EBCDIC]

USE input-file
[RECORD record-type, length, [max-length] [KEY ({start-pos,length,type}...)]
[ORG file-organization]]

[FIELDS ({start-pos,length,[data-type], sort-order}...)
[FORMAT=data-type]
[SKIPREC record-num]]

[SUM [FIELDS] ({start-pos,length,data-type}...)]

[INCLUDE COND ({start-pos,length,data-type,comparison}...)]
[OMIT COND({start-pos,length,data-type,comparison}...)]

GIVE output-file
RECORD record-type,[min-length],max-length [KEY ({start-pos,length,type}...)]
[ORG file-organization]

[ALTSEQ CODE=({old-char1, new-char1}...)]

[INREC [FIELDS] ({start-pos,length,}...),[TRAN=ALTSEC]]

[OUTREC [STARTREC record-1] [ENDREC record-2]
FIELDS][=]([{format-char}]...)
{start-pos, length,{data-format1}TO={data-format2}
[change-nomatch conditions]...},[TRAN=ALTSEC]]

[OUTFIL [FIELDS][=]([{format-char}]...)
{start-pos, length,{data-format1}TO={data-format2}
```

```
[change-nomatch conditions]...},[TRAN=ALTSEC]]

[OUTFIL [STARTREC record-1] [ENDREC record-2]]

[ERROR error-file
RECORD record-type,[min-len],max-len
[KEY ({start-pos,length,type}...)]
[ORG file-organization]
[OUTREC
[FIELDS][=]([{format-char}]...)
{start-pos, length,{data-format1}[TO={data-format2}
[change-nomatch conditions]...},[TRAN=ALTSEC]]]
```

- Format 2

```
citsort take [sort-command-file ]
```

1.3 CitSORT® Options, Statements and Clauses

1.3.1 Options

-test

Verifies syntax alone, without executing the CITSORT command.

As examples:

[No syntax errors reported.]

>citsort -test use presidents.dat record F 85 SORT FIELDS (1,2,NU,D) GIVE PRES5.TXT ORG LS

COBOL-IT Sort/Merge Utility - 2.42

Copyright COBOL-IT S.A.R.L. All rights reserved

Usage of this product is subordinate to a COBOL-IT Enterprise subscription

[A syntax error reported]

C:\COBOL\CobolIT\Samples\sort\presidents>citsort -test use presidents.dat record

F 85 SORT FIELDS (1,2,NU.D) GIVE PRES5.TXT ORG LS

COBOL-IT Sort/Merge Utility - 2.42

Copyright COBOL-IT S.A.R.L. All rights reserved

Usage of this product is subordinate to a COBOL-IT Enterprise subscription

Order must be A for ascending, or D for descending (Got ['NU.D'])

-processmax = <int>

Defines the maximum number of processes to use on a multi-processing machine. This must be defined to enable multi-processing. Note that the –processmax clause must be issued on the command line, even when using a command file.

An example of a usage of the –processmax clause with a command file:

>citsort –processmax=4 take commands.txt

-processrec = <int>

Defines the minimum number of records written before creating a new process. Note that the –processrec clause must be issued on the command line, even when using a command file.

An example of a usage of the –processrec clause with a command file:

>citsort –processrec=10000 take commands.txt

1.3.2 Clauses

SORT clause

Activates the SORT mode.

The SORT mode is the default work mode.

The Input files (USE clauses) are read and sorted into the output files (GIVE clauses) in ASCII order.

SORT EBCDIC clause

Activates the SORT EBCDIC mode.

The Input files (USE clauses) are read and sorted into the output files (GIVE clauses) in EBCDIC order.

MERGE clause

Activate the MERGE mode.

The Input files (USE clauses) are read and written into the output files (GIVE clauses). No sort is performed.

COPY clause

COPY Work Mode is equivalent to MERGE.

SIGN-EBCDIC clause

SIGN-EBCDIC determines the output type of sign in a signed, numeric field. By default, the output is ASCII whether input is ASCII or EBCDIC. Using SIGN-EBCDIC will result in EBCDIC output.

- General Format

```
[SIGN-EBCDIC]
```

- Examples

Placement of the SIGN-EBCDIC clause

```
>CITSORT SIGN-EBCDIC USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D)
GIVE PRES2.TXT ORG LS
```

1.3.3 Statements

ALTSEQ Statement

The ALTSEQ statement is used to change the alternate translation table (ALTSEQ table). Modifications are applied to the standard ASCII/EBCDIC translation table. When using the ALTSEQ statement, the TRAN=ALTSEQ clause must be used in context (any segment list that can include an EDIT or FORMAT clause may include a TRAN=ALTSEQ clause) to apply the modifications described by the ALTSEQ statement.

- General Format

```
[ALTSEQ CODE=({old-char1, new-char1}...)]
```

ALTSEQ Statement used to modify data in copy of existing file

```
>CITSORT FIELDS=COPY USE PRESIDENTS.DAT RECORD F 85 GIVE TESTOUT.DAT
ALTSEQ CODE=(0540,0E40)
OUTREC FIELDS=(1,4,5,TRAN=ALTSEQ)
```

ALTSEQ Statement used with an INREC

```
>CITSORT FIELDS=(1,80,CH,A),SKIPREC=1
USE PRESIDENTS.DAT RECORD F85 GIVE TESTOUT.DAT
INCLUDE COND=(1,80,CH,NE,C'          ')
ALTSEQ CODE=(7F40,6D60)
INREC FIELDS=(1,80,TRAN=ALTSEQ)
SUM  FIELDS=NONE
OUTREC FIELDS=(2,64,16X)
```

ALTSEQ Statement used with an OUTREC

```
>CITSORT FIELDS=COPY
USE PRESIDENTS.DAT RECORD F85 GIVE TESTOUT.DAT
ALTSEQ CODE=(0040,0140,0240,0340,0440,0540,0640,0740,0840,0940)
OUTREC FIELDS=(1,80,TRAN=ALTSEQ)
```

ALTSEQ Statement used with a single value

```
>CITSORT FIELDS=(1,80,CH,A),SKIPREC=1
USE PRESIDENTS.DAT RECORD F85 GIVE TESTOUT.DAT
INCLUDE COND=(1,80,CH,NE,C'          ')
ALTSEQ CODE=(7F40)
INREC FIELDS=(1,80,TRAN=ALTSEQ)
SUM  FIELDS=NONE
OUTREC FIELDS=(2,64,16X)
```

USE Statement

The USE Statement declares the input files, as well as record type, and record length, and the overall organization of the file, whether it is a Sequential, Line Sequential, or Relative file.

- General Format

```
USE input-file
RECORD [record definition]
[ORG   file organization ]
[KEY ({start-pos,length,type}...)]
```

RECORD Clause

The RECORD Clause of the USE Statement describes the record format- fixed-length, or variable- length- and record length of the input file.

- General Format

```
RECORD record-type, length, [max-length]
```

- Syntax
  Record-type is either F (fixed-length) or V (variable-length).

General Rules

- If the RECORD format is variable length, the length parameter holds min-length.
  1. For variable length files, both min-length and max-length are required.
  2. The RECORD Clause is required.

ORG Clause

The ORG Clause of the USE Statement describes the organization- relative, indexed, sequential, or line-sequential of the output file.

- General Format

```
[ORG org-type]
```

- Syntax

Org-type is one of the following:

RL organization is relative

IX organization is indexed (Require a KEY clause) IX1 organization is indexed, and compressed*

SQ organization is sequential

LS organization is line sequential (Linux/Unix record terminators of LF) LSD organization is line sequential (DOS record terminators of CR/LF)**

*When data is compressed in an indexed file, you must describe the organization as ix1, you must declare all indexes.

As an example:

citsort copy

use DATAFIL

org ix1 key=(1,18, P, 291,10, AD) record f 1100 give DATAFILsort

org sq record f 1100

The environment variable CITSORT_LS_DOS forces record terminators for all line sequential files to be CR/LF. For existing installations, where CitSORT is being used in a DOS environment, and where scripts use an **org ls notation, this would be required when updating to version 3.4.18 or later, as the behavior of CITSORT has changed, and by default, the **org ls** notation causes record terminators of LF to be created.

- General Rules

1. The ORG clause is optional.
2. The default value of org-type is sequential

(SQ).

- Examples

USE Statement, Fixed-Length RECORD clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
ORG LS
```

USE Statement, Variable-Length RECORD clause

```
>CITSORT USE PRESIDENTS2.TXT RECORD V 12 24 ORG LS SORT FIELDS (1,2,NU,A) GIVE
PRES3.TXT ORG LS
```

USE Statement, ORGANIZATION Sequential File

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 ORG SQ SORT FIELDS (1,2,NU,D) GIVE
PRES2.TXT ORG LS
```

USE Statement, ORGANIZATION Line Sequential (DOS) File

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 ORG LSD SORT FIELDS (1,2,NU,D) GIVE
PRES2.TXT ORG LSD
```

USE Statement, ORGANIZATION Relative File

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 ORG RL SORT FIELDS (1,2,NU,D) GIVE
PRES2.TXT ORG LS
```

USE Statement, No ORGANIZATION Clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
ORG LS
```

KEY Clause

The KEY Clause of the USE Statement describes the keys of indexed files.

- General Format

```
KEY ({start-pos,length,type}...)]
```

- Syntax

Start-pos is the offset of the beginning of the field, having marked the first byte as offset 1.
Length is the length of the field in bytes

type is one of the following:

P	Primary key
A	Alternate key
AD	Alternate key with duplicates
C	Component of the last-specified primary or alternate key.

- General Rules

1. The KEY clause is required for Indexed files.
2. Primary key must be the first describer key.
3. Primary key may not have duplicates.
4. You must define the keys in order of importance, the primary key first, followed by all its components if it is split, then the first alternate key and all of its components, and so on.

- Examples

Copy the data from PRESIDENT_IX into LINE SEQUENTIAL file PRES2.TXT

```
>CITSORT COPY USE PRESIDENTS_IX RECORD F 85 ORG IX KEY (1,2,P,3,5,AD,10,5,C)
GIVE PRES2.TXT ORG LS
```

PRESIDENT_IX is INDEXED has record of 85 byte and keys are:
Primary: From byte 1 for 2 Byte
Alternate: Split key with duplicates, First part from 3 for 5 byte and second part from 10 for 5 bytes

FIELDS Statement

The FIELDS Statement describes the SORT keys to be used for an input file. SORT keys are described in the FIELDS statement, by their starting position, length, data-type, and sort-order. Multiple SORT keys can be described for a single SORT operation.

- General Format

```
FIELDS ({start-pos,length,[data-type], sort-order}...)[format=data-type][TRAN=ALTSEQ])
```

- Syntax

Start-pos is the offset of the beginning of the field, having marked the first byte as offset 1. Length is the length of the field in bytes

Data-type is a 2-byte abbreviation for a COBOL data type.

- General Rules

1. The 2-byte data-types and corresponding COBOL data types are listed in the data-type table below.

Data-type Abbreviation	COBOL data type

CH	PIC X USAGE DISPLAY
NU	PIC 9 USAGE DISPLAY
PD	PIC S9 COMP-3
FI	PIC S9.99 USAGE DISPLAY
BI	USAGE COMP
C5	USAGE COMP-5
S5	PIC S9 USAGE COMP-5
CX	USAGE COMP-X
LS	PIC S9 LEADING SEPARATE
LI	PIC S9 LEADING INCLUDED
ZD	PIC 9 USAGE DISPLAY
FS	PIC S9.99 USAGE DISPLAY

2. Sort-order is either A (ascending) or D (descending).
3. When the FORMAT clause is used to describe the data-type, the data-type is not included in the FIELDS Clause.

FORMAT Clause

The FORMAT Clause describes the data type of the FIELDS in a SORT Statement.

- General Format

```
[FORMAT=data-type]
```

- Syntax

  Data-type is a 2-byte abbreviation for a COBOL data type.

- General Rules

1. When used, the FORMAT Clause replaces the data-type entry in the FIELDS Clause.
2. The FORMAT Clause is set to a data-type abbreviation (see data type table above).
3. Format may be used when sorting on a single key, but is more useful when sorting on multiple keys that have the same data type.

SKIPREC Clause

The SKIPREC Clause indicates the number of records to skip before beginning to process the SORT.

- General Format

```
[SKIPREC=<number>]
```

- Syntax

<number> is an integer describing the number of records to skip before beginning to process the SORT.

- General Rules

1. When SKIPREC=<number> is indicated, the SORT begins process at record <number> +1.

- Examples

SORT Fields, Sort on Descending, Ascending Order

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
ORG LS
```

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,A) GIVE PRES2.TXT
ORG LS
```

SORT Fields, Usage of the Format Clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,A) FORMAT=NU GIVE
PRES2.TXT ORG LS
```

SORT Fields, Usage of the SKIPREC Clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,A) FORMAT=NU
SKIPREC=1 GIVE PRES2.TXT ORG LS
```

SORT Fields, Sort on Multiple Keys

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (43,21,CH.A,1,2,CH,A) GIVE
PRES2.TXT ORG LS
```

SORT Fields, Usage of the Format Clause on Multiple Key Sort

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (43,21,A,3,22,A) FORMAT=CH
GIVE PRES2.TXT ORG LS
```

INREC Statement

INREC provides a mechanism for taking portions of an input file, and creating a new input file for the SORT function. There could be a number of possible reasons for doing this- but performance is probably the most likely. If an input file is very long, it can be reduced in length, by selecting a subset of its fields, formatting them in a shorter file, and achieving better performance on the SORT. Subsequent SORT FIELDS and SUM FIELDS statements must refer to the offsets created by the INREC FIELDS statement, not to the offsets in the original input file.

INREC also provides an IFTHEN clause which allows you to reformat lines before sorting them according to characteristics detected in your IFTHEN condition.

- General Format

```
[INREC {FIELDS[=]{start-pos,length,}...)]
     [IFTHEN=(WHEN=(start-pos,length,data-type,comparison-expression),
       BUILD=([{format-char}]...)
             {start-pos, length,{data-format1}]
```

The IFTHEN clause, with WHEN and BUILD subclauses are supported in the INREC and OUTREC statements.

As an example:
Consider a case where a line sequential file contains a header and trailer record that have different formats than the main body of the sequential file, for example:

```
hdr 14-jun-2018
04James Madison          180903041817030408Democratic-RepublicanGeorge Clinton
03Thomas Jefferson       180103041809030408Democratic-RepublicanAaron Burr
02John Adams             179703041801030404Federalist           Thomas Jefferson
01George Washington      178904301797030408Independent          John Adams
trl   4 records total
```

The following CITSORT could be used to sort the body of this file, while leaving the header and trailer files in place:

```
USE PRES2.TXT RECORD F 85 ORG LS
INREC IFTHEN=(WHEN=(1,3,CH,EQ,C'hdr'), BUILD=(C'00',1,25)),
IFTHEN=(WHEN=(1,3,CH,EQ,C'trl'), BUILD=(C'05',1,25)),
IFTHEN=(WHEN=(1,2,NU,GE,00), BUILD=(1,85))
SORT FIELDS=(1,2,NU,A)
GIVE PRES3.TXT RECORD F 85 ORG LS
OUTREC IFTHEN=(WHEN=(1,2,CH,EQ,C'00'),BUILD=(3,23)),
IFTHEN=(WHEN=(1,2,CH,EQ,C'05'), BUILD=(3,23))
```

The first INREC/IFTHEN statement creates a condition when character 1-3 are "hdr". In this case, the the characters "00" are inserted at the beginning of the line followed by the characters 1 thru 25 of the line.

The second INREC/IFTHEN statement creates a condition when character 1-3 are "trl". In this case,

the characters "05" are inserted at the beginning of the line followed by the characters 1 thru 25 of the line.

The third INREC/IFTHEN statement creates a condition for the modified body of data. The header file has been marked as record "00". The trailer file as record "05". So the third INREC/IFTHEN condition applies to all lines, identifed with the first two bytes being numeric and GE 0. The lines are re-written as-is.

The file, originally in descending format, is SORTed in ascending format on the numeric fields in bytes 1-2.

The OUTREC/IFTHEN statement strips out the values that have been inserted in character positions 1-2. In records where the first two bytes are '00' or '05', the record is re-built from position 3.

FIELDS Clause

The FIELDS Clause of the INREC Statement iterates the starting position, and length in the input file of the data that will be included in the file input to the sort.

- General Format

```
FIELDS[=]({start-pos,length,}... [TRAN=ALTSEQ])
```

- Syntax
  1. Start-pos is the offset of the beginning of the field, having marked the first byte as offset 1. Length is the length of the field in bytes

- General Rules
  1. Multiple iterations of Start-pos, length are permitted.
  2. The data offsets created by the INREC operation must be used by subsequent SORT, and SUM operations that identify input data by offset.
  3. TRAN=ALTSEQ activates the code modifications described in the ALTSEQ statement.

- Examples

INREC Statement, Reduce the size of the input record to 40 bytes

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (1,40) SORT FIELDS
(1,2,NU,D) GIVE PRES2.TXT ORG LS
```

SUM Statement

The SUM Statement SUMs the indicated fields, and reports the SUMs for all the fields

described, accumulated on the records in the input file that have the same value in the SORT key.

- General Format

```
[SUM FIELDS ({start-pos,length,data-type}...)]
```

FIELDS Clause

The FIELDS Clause of the SUM Statement iterates the starting position, length, and data-types of the fields that are the target of the SUM.

- General Format

```
FIELDS ({start-pos,length,data-type}...[TRAN=ALTSEQ])
```

- Syntax

  Start-pos is the offset of the beginning of the field, having marked the first byte as offset 1. Length is the length of the field in bytes
  Data-type is a 2-byte abbreviation for a COBOL data type.

- General Rules

1. TRAN=ALTSEQ activates the code modifications described in the ALTSEQ statement.
2. The 2-byte data-types and corresponding COBOL data types are listed in the data-type table below.

Data-type Abbreviation	COBOL data type
CH	PIC X USAGE DISPLAY
NU	PIC 9 USAGE DISPLAY
PD	PIC S9 COMP-3
FI	PIC S9.99 USAGE DISPLAY
BI	USAGE COMP
C5	USAGE COMP-5
S5	PIC S9 USAGE COMP-5
CX	USAGE COMP-X
LS	PIC S9 LEADING SEPARATE
LI	PIC S9 LEADING INCLUDED
ZD	PIC 9 USAGE DISPLAY
FS	PIC S9.99 USAGE DISPLAY

1. Only numeric data types may be named in the FIELDS Clause of a SUM Statement.
2. The SUM statement eliminates records with duplicate sort keys from the output file.
3. The fields preserved in the output record are taken from the first record in which the sort key is found.
4. When an INREC Statement has been used, the offsets used in the SUM FIELDS clause must refer to the offsets of file described by the INREC Clause, not the original input file.
5. When an INREC Statement is used, the SUM Statement must be placed after the INREC Statement.
6. When SUM Fields is set to (None), the effect is to eliminate all records that have duplicate Sort Keys from the output file, preserving only the first instance.
7. The SUM Statement may be placed before or after the GIVE

Statement. Examples

SUM, Summing on a numeric fieldmeric field
This report will list all of the political parties, and the total number of years that Presidents from each of the Parties have served.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (43,21,41,2) SORT FIELDS
(1,21,CH,A) SUM FIELDS=(22,2,NU) GIVE PRES2.TXT ORG LS
```

SUM, Eliminating records with a duplicate sort key

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (3,21,CH,A) GIVE PRES2.TXT
ORG LS SUM FIELDS=(NONE)
```

INCLUDE Statement

The INCLUDE Statement describes a sort filter with a portion of the input file, and a comparison expression. If the portion of the file tests true for the comparison, the record is included in the output file.

General Format

```
[INCLUDE COND ({start-pos,length,data-type,comparison-expression}...)]
```

COND Clause

The COND Clause defines the condition that is applied to determine if the record is included in the output file.

- General Format

```
COND ({start-pos,length,data-type,comparison-expression}...)
```

- Syntax

Start-pos is the offset of the beginning of the field, having marked the first byte as offset 1. Length is the length of the field in bytes
Data-type is a 2-byte abbreviation for a COBOL data type.

- General Rules

1. The 2-byte data-types supported for comparison and corresponding COBOL data types are listed in the data-type table below.

Data-type Abbreviation	COBOL data type
CH	PIC X USAGE DISPLAY
NU	PIC 9 USAGE DISPLAY
PD	PIC S9 COMP-3
FI	PIC S9.99 USAGE DISPLAY
BI	USAGE COMP
C5	USAGE COMP-5
S5	PIC S9 USAGE COMP-5
CX	USAGE COMP-X
LS	PIC S9 LEADING SEPARATE
LI	PIC S9 LEADING INCLUDED
ZD	PIC 9 USAGE DISPLAY
FS	PIC S9.99 USAGE DISPLAY

2. Comparison-expression is minimally made up of a ***comparison-operator***, and ***comparison-operand.***
3. Multiple comparison-operator/comparison-operand pairs may be grouped together with the ***logical-operators*** AND, and OR, as described below.
4. Comparison-operator is one of the following:

EQ	Equal to
NE	Not Equal to
GT	Greater Than
GE	Greater Than or Equal
LT	Less Than
LE	Less Than or Equal

5. Comparison-operand is data being examined.
6. Character data must be marked with single-quotes.
7. Logical-**operators** AND, OR, are supported.
8. AND statements are evaluated before OR statements.
9. In comparison expressions, numeric types may be compared against each other or against a constant.
10. Strings may be compared against each other, a string, or a hexadecimal constant.
11. Substrings may be included in sort filters, but may only be used with the comparison operators “EQ” (Equal) and “NE” (Not Equal).
12. Data types BI, C5, C6, CX, LI, LS, NU, PD, S5, SB and ZD may be compared against each other, or against a hexadecimal or decimal constant.
13. BI and CH may be compared against each other or a string constant.
14. CH may be compared against a hexadecimal constant.

Examples

INCLUDE Statement, Character comparison
This example shows the application of a character sort filter, which will only retrieve Presidents, whose first names begin with “John”.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D)  INCLUDE
COND=(3,4,CH,EQ,'JOHN')GIVE PRES2.TXT ORG LS
```

NCLUDE Statement, Numeric comparison
This example shows the application of a numeric sort filter, which will only retrieve President number 10.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (1,40) SORT FIELDS
(1,2,NU,D) INCLUDE COND=(1,2,NU,EQ,10)GIVE PRES2.TXT ORG LS
```

INCLUDE Statement, Data expressed as hexadecimal

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (1,40) SORT FIELDS
(1,2,NU,D) INCLUDE COND=(1,2,NU,EQ,X’3130’)GIVE PRES2.TXT ORG LS
```

INCLUDE Statement, Use of Logical AND operator
This example contains a logical AND operator. It will retrieve a list of Presidents numbers greater than 1 and less than 10.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) INCLUDE
COND=(1,2,NU,GT,01,AND,1,2,NU,LT,10) GIVE PRES2.TXT ORG LS
```

INCLUDE Statement, Use of Logical OR operator
This example contains a logical OR operator. It will retrieve a list of Presidents

numbers equal 1 or equal 10.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) INCLUDE
COND=(1,2,NU,EQ,01,OR,1,2,NU,EQ,10)GIVE PRES2.TXT ORG LS
```

INCLUDE Statement, Use of Substring operator
This example includes records in which the substring 'John' has been found inside a 21-byte span of characters in the record. Note- The substring function can only be used with the "eq" (equal) and "ne" (not equal) comparison operators.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) INCLUDE
COND=(3,21,SS,EQ,'JOHN')GIVE PRES2.TXT ORG LS
```

OMIT Statement

The OMIT Statement describes a sort filter with a portion of the input file, and a comparison expression. If the portion of the file tests true for the comparison, the record is omitted from the output file.

- General Format

```
[OMIT COND ({start-pos,length,data-type,comparison-expression}...)]
```

COND Clause

The COND Clause defines the condition that is applied to determine if the record is omitted from the output file.

- General Format

```
COND ({start-pos,length,data-type,comparison-expression}...)
```

- Syntax

  Start-pos is the offset of the beginning of the field, having marked the first byte as offset 1. Length is the length of the field in bytes
  Data-type is a 2-byte abbreviation for a COBOL data type.

- General Rules

1. The 2-byte data-types supported for comparison and corresponding COBOL data types are listed in the data-type table below.

Data-type Abbreviation	COBOL data type
CH	PIC X USAGE DISPLAY
NU	PIC 9 USAGE DISPLAY
PD	PIC S9 COMP-3
FI	PIC S9.99 USAGE DISPLAY
BI	USAGE COMP
C5	USAGE COMP-5
S5	PIC S9 USAGE COMP-5
CX	USAGE COMP-X
LS	PIC S9 LEADING SEPARATE
LI	PIC S9 LEADING INCLUDED
ZD	PIC 9 USAGE DISPLAY
FS	PIC S9.99 USAGE DISPLAY

2. Comparison-expression is minimally made up of a ***comparison-operator***, and ***comparison- operand.***
3. Multiple comparison-operator/comparison-operand pairs may be grouped together with the
   logical-operators AND, and OR, as described below.
4. Comparison-operator is one of the following:

EQ	Equal to
NE	Not Equal to
GT	Greater Than
GE	Greater Than or Equal
LT	Less Than
LE	Less Than or Equal

5. Comparison-operand is data being examined.
6. Character data must be marked with single-quotes.
7. Logical**-operators** AND, OR, are supported.
8. AND statements are evaluated before OR statements.
9. In comparison expressions, numeric types may be compared against each other or against a constant.
10. Strings may be compared against each other, a string, or a hexadecimal constant.
11. Substrings may be included in sort filters, but may only be used with the comparison operators “EQ” (Equal) and “NE” (Not Equal).
12. Data types BI, C5, C6, CX, LI, LS, NU, PD, S5, SB, and ZD may be compared against each other, or against a hexadecimal or decimal constant.
13. BI and CH may be compared against each other or a string constant.
14. CH may be compared against a hexadecimal

constant.

Examples

OMIT Statement, Character Comparison

This example shows the application of a character sort filter, which will omit Presidents, whose first names begin with “John”.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) OMIT
COND=(3,4,CH,EQ,'JOHN')GIVE PRES2.TXT ORG LS
```

OMIT Statement, Numeric Comparison

This example shows the application of a numeric sort filter, which will only omit President number 10.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (1,40) SORT FIELDS
(1,2,NU,D) OMIT COND=(1,2,NU,EQ,10)GIVE PRES2.TXT ORG LS
```

OMIT Statement, Data expressed as hexadecimal

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (1,40) SORT FIELDS
(1,2,NU,D) OMIT COND=(1,2,NU,EQ,X’3130’)GIVE PRES2.TXT ORG LS
```

OMIT Statement, Usae of Logical AND operator

This example shows the application of a numeric sort filter that contains a logical AND operator. It will omit Presidents with numbers greater than 1 and less than 10.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) OMIT
COND=(1,2,NU,GT,01,AND,1,2,NU,LT,10)GIVE PRES2.TXT ORG LS
```

Omit Statement, Usage of Logical OR operator

This example shows the application of a numeric sort filter that contains a logical OR operator. It will retrieve a list of Presidents omitting numbers equal 1 or equal 10.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) OMIT
COND=(1,2,NU,EQ,01,OR,1,2,NU,EQ,10) GIVE PRES2.TXT ORG LS
```

OMIT Statement, Use of Substring operator

This example omits records in which the substring 'John' has been found inside a 21-byte span of characters in the record. Note- The substring function can only be used with the "eq" (equal) and "ne" (not equal) comparison operators.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) OMIT
COND=(3,21,SS,EQ,'JOHN')GIVE PRES2.TXT ORG LS
```

Error Statement

The ERROR Statement declares the output files that will be written with record that could not be written in GIVE output file because of Invalid or Duplicated key.

Formal syntax is the same as GIVE statement.

See GIVE statement for syntax detail.

- General Format

```
[ERROR error-file
RECORD record-type,[min-length],max-length
   [KEY ({start-pos,length,type}...)]
[ORG file-organization]
[OUTREC [FIELDS][=]([{format-char}]...)
{start-pos, length,{data-format1}[TO={data-format2}]

[change-nomatch conditions]...}]]
```

- General Rules

To avoid cyclic duplicate key error, use only SQ, RL or LS organization for error files

GIVE Statement

The GIVE Statement declares the output files, as well as record type, and record length, and the overall organization of the file, whether it is a Sequential, Line Sequential, or Relative file.

- General Format

```
GIVE output-file
RECORD [record definition]
[ORG   file organization ]
[KEY ({start-pos,length,type}...)]
[FILLBYTE int <0-255> ]
```

RECORD Clause

The RECORD Clause of the GIVE Statement describes the record format- fixed-length, or variable- length- and record length of the output file.

- General Format

```
RECORD record-type, length, [max-length]
```

- Syntax

  Record-type is either F (fixed-length) or V (variable-length).

- General Rules

  1. If the RECORD format is variable length, both length and max-length are required.
  2. The RECORD Clause is optional, and if not included, will be the same as the USE file.

ORG Clause

The ORG Clause of the GIVE Statement describes the organization- relative, indexed, sequential, or line-sequential of the output file. The ORG clause is optional. The default is sequential.

- General Format

```
[ORG org-type]
```

- Syntax

Org-type is one of the following:

RL organization is relative

IX organization is indexed (Require a KEY clause) IX1 organization is indexed, and compressed*

SQ organization is sequential

LS organization is line sequential (Linux/Unix record terminators of LF) LSD organization is line sequential (DOS record terminators of CR/LF)**

*When data is compressed in an indexed file, you must describe the organization as ix1, and you must declare all indexes. As an example:

citsort copy

use DATAFIL

org ix1 key=(1,18, P, 291,10, AD) record f 1100 give DATAFILsort

org sq record f 1100

The environment variable CITSORT_LS_DOS forces record terminators for all line sequential files to be CR/LF. For existing installations, where CitSORT is being used in a DOS environment, and where scripts use an **org ls notation, this would be required when updating to version 3.4.18 or later, as the behavior of CITSORT has changed, and by default, the **org ls** notation causes record terminators of LF to be created.

- General Rules

1. The default value of org-type is SQ.
2. When using IX (Index files) Record that generate duplicate key or Invalid key error will be written in the ERROR file if defined.

- Examples

GIVE Statement, Fixed-Length RECORD clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
RECORD F 85 ORG LS
```

GIVE Statement, Organization Line Sequential

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
ORG LS
```

KEY Clause

The KEY Clause of the GIVE Statement describes the keys of indexed files.

- General Format

```
KEY ({start-pos,length,type}...)]
```

- Syntax

Start-pos is the offset of the beginning of the field, having marked the first byte as offset 1.
Length is the length of the field in bytes

type is one of the following:

P	Primary key
A	Alternate key
AD	Alternate key with duplicates
C	Component of the last-specified primary or alternate key.

- General Rules

5. The KEY clause is required for Indexed files.
6. Primary key must be the first describer key.
7. Primary key may not have duplicates.
8. You must define the keys in order of importance, the primary key first, followed by all its components if it is split, then the first alternate key and all of its components, and so on.

- Examples

```
>CITSORT COPY USE PRESIDENTS_IX RECORD F 85 ORG IX KEY (1,2,P,3,5,AD,10,5,C)
GIVE PRES2.TXT ORG LS
```

Copy the data of the file PRESIDENT_IX into the LINE SEQUENTIAL file PRES2.TXT. PRESIDENT_IX is INDEXED has record of 85 byte and keys are:

Primary: From byte 1 for 2 Byte
Alternate: Split key with duplicates, First part from 3 for 5 byte and second part from 10 for 5 bytes

FILLBYTE Clause

The FILLBYTE Clause of the GIVE Statement describes the character used to pad trailing spaces in a sequential file.

- General Format

```
FILLBYTE (INT 0-255)
```

- Syntax

INT is the ASCII value of the FILLBYTE character.

- General Rules

5. When FILLBYTE is specified, the output record will be padded (if needed) with that byte value instead of space.

- Examples

```
> CITSORT COPY USE PRESIDENTS2.TXT ORG LS RECORD V 12,24 GIVE
PRESIDENTS3.TXT ORG SQ RECORD F 85 FILLBYTE 42
```

OUTFIL Statements

If you wish to produce more than one output file, with different output and different formats, you can associate an **outfil** statement with a **give** statement, and then describe the reformatting associated with the **OUTREC** statement. You can repeat this process multiple times to produce multiple output files, each of which is formatted differently.

- General Format

```
[OUTFIL]
[ STARTREC record-number-1 ]
[ ENDREC record-number-2 ]
[OUTREC [FIELDS][=]([{format-char}]...)
{start-pos, length,{data-format1}[TO={data-format2}]
[change-nomatch conditions]...}]
```

STARTREC Clause

After a fixed length file has been sorted, the OUTFIL STARTREC clause can be used to indicate that the output file should begin at an offset in the record that is determined by record-number-1. Record-number-1 is the first record that is printed in the output file.

- General Format

```
[ STARTREC record-number-1 ]
```

- Syntax

  Record-number-1 is a numeric data item or literal.

- General Rules

  1. Record-number-1 is the first record that is printed in the output file.
  2. The STARTREC Clause must be placed after an OUTFIL Clause.

ENDREC Clause

After a fixed length file has been sorted, the OUTFIL ENDREC Clause can be used to indicate that the output file should end at an offset in the record that is determined by record-number-2.

- General Format

```
[ ENDREC record-number-2 ]
```

- Syntax

  Record-number-2 is a numeric data item or literal.

General Rules

1. Record-number-2 is the last record that is printed in the output file.
2. The ENDREC Clause must be placed after an OUTFIL Clause.

Examples

OUTFIL, Usage of the STARTREC Clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,D) FORMAT=NU GIVE
PRES2.TXT ORG LS OUTFIL STARTREC 40
```

OUTFIL, Usage of the ENDREC Clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,D) FORMAT=NU GIVE
PRES2.TXT ORG LS OUTFIL ENDREC 10
```

OUTFIL, Multiple output files, Multiple OUTRECs

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE OUT1.TXT
OUTFIL OUTREC=(1,2,' ',3,22,' ',43,21) GIVE OUT2.TXT ORG LS OUTFIL
OUTREC=(3,21,' ',25,4,'-',33,4)
```

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE OUT1.TXT
ORG LS OUTFIL STARTREC=40 OUTREC=1,2,' `,3,22,' `, 43,21) GIVE OUT2.TXT ORG LS
OUTFIL ENDREC=5 OUTREC=(3,21,' `, 25,4,'-`,33,4)
```

OUTREC Statement

OUTREC provides a mechanism for reformatting an output file, using portions of the output file that have been created by the SORT. This functionality can be useful, as it allows the user to select which fields they would like to see in the output file, and in what order they would like to see them.

OUTREC also provides an IFTHEN clause which allows you to reformat lines based on conditions detected in your IFTHEN condition.

- General Format

```
[OUTREC [FIELDS][=]([[{format-char}]...)
{start-pos, length,{data-format1}[TO=data-format2]}
[change-nomatch conditions]...)]]
[FILLBYTE int <0-255> ]
[IFTHEN=(WHEN=(start-pos,length,data-type,comparison-expression),
BUILD=([{format-char}]...){start-pos, length, {data-format 1 } ]
```

The IFTHEN clause, with WHEN and BUILD subclauses are supported in the INREC and OUTREC statements.

As an example:
Consider a case where a line sequential file contains a header and trailer record that have different formats than the main body of the sequential file, for example:

```
hdr 14-jun-2018
04James Madison         180903041817030408Democratic-RepublicanGeorge Clinton
03Thomas Jefferson      180103041809030408Democratic-RepublicanAaron Burr
02John Adams            179703041801030404Federalist           Thomas Jefferson
01George Washington     178904301797030408Independent          John Adams
trl   4 records total
```

The following CITSORT could be used to sort the body of this file, while leaving the header and trailer files in place:

```
USE PRES2.TXT RECORD F 85 ORG LS
INREC IFTHEN=(WHEN=(1,3,CH,EQ,C'hdr'), BUILD=(C'00',1,25)),
IFTHEN=(WHEN=(1,3,CH,EQ,C'trl'), BUILD=(C'05',1,25)),
IFTHEN=(WHEN=(1,2,NU,GE,00), BUILD=(1,85))
SORT FIELDS=(1,2,NU,A)
GIVE PRES3.TXT RECORD F 85 ORG LS
OUTREC IFTHEN=(WHEN=(1,2,CH,EQ,C'00'),BUILD=(3,23)),
IFTHEN=(WHEN=(1,2,CH,EQ,C'05'), BUILD=(3,23))
```

The first INREC/IFTHEN statement creates a condition when character 1-3 are "hdr". In this case, the the characters "00" are inserted at the beginning of the line followed by the characters 1 thru 25 of the line.

The second INREC/IFTHEN statement creates a condition when character 1-3 are "trl". In this case, the characters "05" are inserted at the beginning of the line followed by the characters 1 thru 25 of

the line.

The third INREC/IFTHEN statement creates a condition for the modified body of data. The header file has been marked as record "00". The trailer file as record "05". So the third INREC/IFTHEN condition applies to all lines, identifed with the first two bytes being numeric and GE 0. The lines are re-written as-is.

The file, originally in descending format, is SORTed in ascending format
on the numeric fields in bytes 1-2.

The OUTREC/IFTHEN statement strips out the values that have been inserted in character positions 1-2. In records where the first two bytes are '00' or '05', the record is re-built from position 3.

FIELDS Clause

The FIELDS Clause of the OUTREC Statement describes the format of the output file. Formatting capabilities include the ability to add character, change the order of fields, and replace all instances of a given character string with another character string.

- General Format

```
[FIELDS][=]([{format-char}]...)
{start-pos, length,{data-format1}[TO={data-format2}]
[change-nomatch conditions]...}[TRAN=ALTSEQ]]
```

- Syntax

1. Format-char is a character string surrounded by single quotes.
2. Start-pos is the offset of the beginning of the field, having marked the first byte as offset 1.
3. Length is the length of the field in bytes.
4. Data-format1 is a data format supported by the FIELD statement.
5. Data-format2 is a data format supported by the FIELD statement.

- General Rules

1. 1. TRAN=ALTSEQ activates the code modifications described in the ALTSEQ statement.
2. Multiple iterations of Start-pos, length are permitted.
3. TO= causes a data conversion to be performed on the output file from data-format1 to data-format2.
2. Change-nomatch conditions can be entered after an iteration of start-pos, length, to scan those bytes for certain values and change them to other values.

CHANGE Clause

The Change Clause describes a scan-string, and a replacement string.

- General Format

```
[ CHANGE=({change-length, scan-string, change-string}... ) ]
```

- Syntax

  Change-length is the number of bytes that are needed to accommodate the change. Scan-string is the byte, or bytes which are changed when found in the string.

  Change-string is the string that will be used to replace scan-string in the output record.

- General Rules

  1. Change-length must be long enough to accommodate the change-string.
  2. The substrings (or characters if the substring has a length of one byte) used in a the scan-string and change-string clauses must be delineated by single quotation marks.

NOMATCH Clause

The Nomatch Clause stipulates that if a Change condition is not encountered, the bytes identified by the starting-position and length should be preserved in the output file.

- General Format

```
[ NOMATCH=(start-pos, length ) ]
```

- Syntax

  Start-pos is the offset of the beginning of the field, having marked the first byte as offset 1. Length is the length of the field in bytes

- General Rules

  1. Bytes identified in tihe NOMATCH Clause will not be changed if they do not

match the corresponding CHANGE condition.

2. If there is no NOMATCH Clause, bytes that do not match the corresponding CHANGE condition are initialized to spaces.

- Examples

OUTREC Statement, Reformatting an output record

This example shows how an OUTREC statement can be used to insert formatting characters into an output file using single quotes, as well as how the output file can be reformatted using the starting-position, length indicators.

```
> CITSORT USE PRESIDENTS.DAT ORG SQ RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE
PRES3.TXT ORG LS OUTREC FIELDS=('PRESIDENT # ',1,2,' ',3,21)
```

OUTREC Statement, Reformatting output record, converting data with TO=

This sample illustrates syntax for converting a numeric data item to FS data format using the TO= clause.

```
> CITSORT USE PRESIDENTS.DAT ORG SQ RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE
PRES3.TXT ORG LS OUTREC FIELDS=(1,2,NU,TO=FS,' ',3,21)
```

OUTFILE, OUTREC using CHANGE/NOMATCH

This sample changes 1 byte, at position 3 from ‘B’ to ‘W’. Otherwise, the existing data is preserved.

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
ORG LS OUTFIL OUTREC=(1,2,3,1,CHANGE=(1,'B','W',NOMATCH=(3,1)),4,36)
```

FILLBYTE Clause

The FILLBYTE Clause of the OUTREC Statement describes the character used to pad trailing spaces in a sequential file.

- General Format

```
FILLBYTE (INT 0-255)
```

- Syntax

INT is the ASCII value of the FILLBYTE character.

- General Rules

5. When FILLBYTE is specified, the output record will be padded (if needed) with that byte value instead of space.

- Examples

```
> CITSORT USE PRESIDENTS.DAT ORG SQ RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE
PRES3.TXT ORG LS OUTREC FIELDS=('PRESIDENT # ',1,2,' ',3,21), FILLBYTE 42
```

SORT-EBCDIC functionality

The SORT-EBCDIC function causes record-sequential ASCII files to be sorted in EBCDIC order.

- General Format

```
SORT-EBCDIC
FIELDS ({start-pos,length,[data-type], sort-order}...) [FORMAT=data-
type]
USE input-file-1 RECORD [record definition] ORG [file organization]
USE input-file-2 . . .
GIVE output-file RECORD [record definition] ORG [file organization]
```

- General Rules

1. The RECORD and ORG Clauses are required for the first input file.
2. Subsequent input files are required to have the same Format and Organization, so this information does not have to be repeated.

- Examples

Sort a file in EBCDIC order:

```
>CITSORT SORT-EBCDIC FIELDS (1,2,NU,D) USE PRESIDENTS.DAT RECORD F 85  GIVE
BBB.TXT ORG LSD
```

CitSORT MERGE functionality

The Merge function can be used to combine two or more files that have identical record formats, and have both been Sorted on the same key, or keys.

- General Format

```
USE input-file-1 RECORD [record definition] ORG [file organization]
USE input-file-2 . . .
SORT FIELDS ({start-pos,length,[data-type], sort-order}...)
[FORMAT=data-type]
GIVE output-file RECORD [record definition] ORG [file organization]
```

- General Rules

3. The RECORD and ORG Clauses are required for the first input file.
4. Subsequent input files are required to have the same Format and Organization, so this information does not have to be repeated.

- Examples

Merge two files sorted on the same key into a single file

```
>CITSORT USE PRESMERGE1.TXT RECORD F 85 ORG LS USE PRESMERGE2.TXT SORT
FIELDS=(1,2,NU,A) GIVE ALLPRES.TXT ORG LS
```

CitSORT COPY functionality

The Copy function creates a new data set which is the duplicate of the original data set.

- General Format

```
[Fields=]COPY
```

- General Rules

1. The file named in the USE clause is duplicated in the file named in the GIVE clause.
2. Multiple GIVE clauses are allowed, allowing the COPY function to duplicate a single file in multiple files.

- Examples

To create a copy of an existing file

```
>CITSORT FIELDS=COPY USE PRESIDENTS.DAT RECORD F 85 GIVE TESTOUT.DAT
```

To create multiple copies of an existing file

```
>CITSORT COPY USE PRESIDENTS.DAT RECORD F 85 GIVE TESTOUT.DAT GIVE
TESTOUT2.DAT
```

To convert a fixed length C-ISAM file to a COBOL-IT VBISAM file

A fixed length C-ISAM file is basically a sequential file with a linefeed (x"0A") character added to the end of each record. When a record is deleted the linefeed character is replaced with a x"00" character.

The command file shown below declares the C-ISAM file as a sequential file with a length of 40. That's the ISAM record length of 39 plus the ending character. It then uses "OMIT COND" to detect deleted records by checking for the x"00" character at the end of each record. It declares the COBOL-IT ISAM file with a key of the first 9 bytes of the record.

```
>CITSORT COPY USE
CISAMFILE.DAT ORG
SQ
RECORD F 40
OMIT COND=(40,1,CH,EQ,X'00')
INREC FIELDS (1,39)
GIVE VBISAMFILE F 39
ORG IX
KEY (1,9,P)
ERROR VGCITERRORFILE.TXT ORG SQ
```

Sorting a C-Tree File using EXTFH

The Use File and Give File are both indexed files

The EXTFH driver for C-Tree indexed files must be used to operate the sort on the C-Tree file that is described in the USE and GIVE clauses. Set the following environment variables:

COB_CTREE_PATH=C:\FairCom\V10.4.0.RTG\winX64\Driver\ctree.cobol\EXTFH

COB_EXTFH=CTEXTFH

COB_EXTFH_LIB=C:\FairCom\V10.4.0.RTG\winX64\Driver\ctree.cobol\EXTFH\CTEXTFH.dll;$ COBOLITDIR\bin\libcitextfh_dll.dll

Then, run your command:

citsort use ABTL ORG IX KEY=(3,4,P,3,2,AD,49,1,C) RECORD F 282 SORT FIELDS (7,10,CH,A) GIVE ausg.txt ORG IX

Dumping a C-Tree File into a Line Sequential File using EXTFH

The Use File is an indexed file and the Give file is a Line Sequential file

The EXTFH driver for C-Tree indexed files must be used to open and read the C-Tree file that is described in the USE clause. The COB_EXTFH_LIB and the EXTFH driver for Flat Files must be used to write the Line Sequential file in the GIVE clause. Set the following environment variables:

COB_CTREE_PATH=C:\FairCom\V10.4.0.RTG\winX64\Driver\ctree.cobol\EXTFH

COB_EXTFH_INDEXED=CTEXTFH

COB_EXTFH_FLAT=EXTFH

COB_EXTFH_LIB=C:\FairCom\V10.4.0.RTG\winX64\Driver\ctree.cobol\EXTFH\CTEXTFH.dll;$ COBOLITDIR\bin\libcitextfh_dll.dll

Then, run your command:

citsort use ABTL ORG IX KEY=(3,4,P,3,2,AD,49,1,C) RECORD F 282 SORT FIELDS (7,10,CH,A) GIVE out.txt ORG LS

CitSORT TAKE [command-file]

CitSORT **take [command-file]** provides the convenience of storing CitSORT commands in a text file. The TAKE Statement allows for the storage of CitSORT commands in a text file, in

which they can be formatted with comments, and more easily maintained.

- General Format

```
take [ sort-command-file ]
```

- Syntax

Sort-command-file is a text file that contains CitSORT commands.

- General Rules

1. The CitSORT commands in sort-command-file may contain comments, which are preceded by an "*" asterisk.
2. The CitSORT commands may be separated by line terminators (LF, or CR/LF).

- Examples

CitSORT TAKE functionality

```
>CITSORT TAKE PRESIDENT-PARAMS.TXT
```

As an example president-params.txt contains:

```
Use presidents.dat          * identifies the file to be sorted.
Record F 85                 * Fixed length, 85 bytes, line seq
Sort Fields (1,2,nu,d)      * sort key bytes 1-2/descending
order Give pres2.txt org ls * output to pres2.txt, line seq
```

2 Appendices

2.1 Presidents.dat, Presidents2.dat

The file presidents.dat, used in many examples in this document, is a fixed-length sequential file with a record length of 85 bytes. As a fixed-length sequential file, it can be described as either as sequential file (org sq) or a relative file (org rl). The data in presidents.dat is below.

```
01George Washington 178904301797030408Independent John Adams 02John
Adams 179703041801030404Federalist Thomas Jefferson 03Thomas Jefferson
180103041809030408Democratic-RepublicanAaron Burr 04James Madison
180903041817030408Democratic-RepublicanGeorge Clinton 05James Monroe
181703041825030408Democratic-RepublicanDaniel D. Tompkins 06John Quincy Adams
182503041829030404Democratic-RepublicanJohn C. Calhoun 07Andrew Jackson
182903041837030408Democratic John C. Calhoun 08Martin Van Buren
183703041841030404Democratic Richard Mentor Johnson09William Henry
Harrison184103041841040400Whig John Tyler 10John Tyler
184104041845030404Whig Vacant 11James K. Polk
184503041849030404Democratic George M. Dallas 12Zachary Taylor
184903041850070901Whig Millard Fillmore 13Millard Fillmore
185007091853030403Whig Vacant 14Franklin Pierce
185303041857030404Democratic William R. King 15James Buchanan
185703041861030404Democratic John C. Breckinridge 16Abraham Lincoln
186103041865041504Republican Andrew Johnson 17Andrew Johnson
186504151869030404Democratic Vacant 18Ulysses S. Grant
186903041877030408Republican Schuyler Colfax 19Rutherford B. Hayes
187703041881030404Republican William A. Wheeler 20James A. Garfield
188103041881091900Republican Chester A. Arthur 21Chester A. Arthur
188109191885030404Republican Vacant 22Grover Cleveland
188503041889030404Democratic Thomas A. Hendricks 23Benjamin Harrison
188903041893030404Republican Levi P. Morton 24Grover Cleveland
189303041897030404Democratic Adlai E. Stevenson 25William McKinley
189703041901091404Republican Theodore Roosevelt 26Theodore Roosevelt
190109141909030408Republican Vacant 27William Howard Taft
190903041913030404Republican James S. Sherman 28Woodrow Wilson
191303041921030408Democratic Thomas R. Marshall 29Warren G. Harding
192103041923080202Republican Calvin Coolidge 30Calvin Coolidge
192308021929030407Republican Charles G. Dawes 31Herbert Hoover
192903041933030404Republican Charles Curtis 32Franklin D. Roosevelt
193303041945041212Democratic Harry S. Truman 33Harry S Truman
194504121953012008Democratic Alben W. Barkley 34Dwight D. Eisenhower
195301201961012008Republican Richard Nixon 35John F. Kennedy
196101201963112202Democratic Lyndon B. Johnson 36Lyndon B. Johnson
196311221969012006Democratic Hubert Humphrey 37Richard Nixon
196901201974080905Republican Gerald Ford 38Gerald Ford
197408091977012003Republican Nelson Rockefeller 39Jimmy Carter
```

```
19770120198101200 4Democratic      Walter Mondale      40Ronald Reagan
19810120198901200 8Republican      George H. W. Bush   41George H. W. Bush
19890120199301200 4Republican      Dan Quayle          42Bill Clinton
19930120200101200 8Democratic      Al Gore             43George W. Bush
20010120200901200 8Republican      Dick Cheney         44Barack Obama
20090120201301200 4Democratic      Joe Biden
```

The file presidents2.txt is a variable-length line sequential file with a minimum record length of 12 bytes and a maximum record length of 24 bytes. The data in presidents2.txt is below:

```
44Barack Obama
43George W. Bush
42Bill Clinton
41George H. W. Bush
40Ronald Reagan
39Jimmy Carter
38Gerald Ford
37Richard Nixon
36Lyndon B. Johnson
35John F. Kennedy
34Dwight D. Eisenhower
33Harry S Truman
33Warren G. Harding
32Franklin D. Roosevelt
31Herbert Hoover
30Calvin Coolidge
28Woodrow Wilson
27William Howard Taft
26Theodore Roosevelt
25William McKinley
24Grover Cleveland
23Benjamin Harrison
22Grover Cleveland
21Chester A. Arthur
20James A. Garfield
19Rutherford B. Hayes
18Ulysses S. Grant
17Andrew Johnson
16Abraham Lincoln
15James Buchanan
14Franklin Pierce
13Millard Fillmore
12Zachary Taylor
11James K. Polk
10John Tyler
09William Henry Harrison
```

08Martin Van Buren
07Andrew Jackson
06John Quincy Adams
05James Monroe
04James Madison
03Thomas Jefferson
02John Adams
01George Washington

2.2 Performance Matters

SORTs of very large files often leave heavy footprints on the overall performance of an application. So much so, that this is often the first area examined, when addressing the imperative of improving performance.

CitSORT is fast, typically exceeding expectations in the area of performance. Below, we've constructed a test where we see a simple SORT of a 1 million-record sequential file being reduced from 46 seconds to 5 seconds using CitSORT.

*

A test- Create a sequential file with 1,000,000 records, and use CitSORT to reverse the order of the records.

1- Create the file to be sorted with the following code.

```
*
identification
division. program-
id.  labfile2.
environment
division. input-
output section.
file-control.
    select lab2fil assign to "lab2recs.txt"
    organization is sequential
    access is
    sequential file
    status is file-stat.
data
division.
file
section.
fd lab2fil.
01 lab2fil-record.
    05 sequence-no     pic
    9(7). 05 name    pic
```

```
    X(25).
    05 addr          pic X(25).
    05 zip           pic 9(10).
    05 salary        pic 9(7)V99.
*
 working-storage
 section. 77 ws-
 dummy            pic
 x.
 77  file-stat  pic
 xx.    procedure
 division. main.
    initialize sequence-no.
    move all "A" to name.
    move all "B" to addr.
    move 1234567890 to
    zip.
    move 9876543.21 to salary.
*
    open output lab2fil.
    perform load-file 1000000 times.
    close lab2fil.
    stop run.
*
 load-file.
    add 1 to sequence-
    no. write lab2fil-
    record.
*    display sequence-no line 10 col 10.
```

2- Sort the file, reversing the order of the records, and record the time it takes with the following batch file:

Lab2a.bat

```
@echo off
now > begin2a.txt
citsort use lab2recs.txt record f 76 sort fields=(1,7,nu,d) give lab2citsort.txt
now >> begin2a.txt
type begin2a.txt
```

4- Compare this with the performance achieved using the SORT verb in a COBOL program to perform the same function:
5- Use the following COBOL program for purposes of comparison:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. lab2.
ENVIRONMENT
DIVISION. INPUT-
OUTPUT SECTION.
FILE-CONTROL.

*SORT input file

    SELECT lab2fil assign to
    "lab2recs.txt" ORGANIZATION IS
    SEQUENTIAL ACCESS IS
    SEQUENTIAL

    FILE STATUS is infile-stat.


*SORT output file

    SELECT sortout assign to
    "testout.txt" organization is
    SEQUENTIAL ACCESS IS
    SEQUENTIAL

    FILE STATUS IS outfile-stat.


*sort file (SD)

    SELECT member-sort ASSIGN TO "sort-work".


DATA
DIVISION. FILE
SECTION.

FD lab2fil.

01 lab2fil-record.

    05 sequence-no     pic
    9(7). 05 name    pic
    X(25).

    05 addr          pic X(25).

    05 zip           pic 9(10).

    05 salary         pic 9(7)V99.


FD sortout.

01 sortout-record.
```

```
           05 out-sequence-no     pic
           9(7). 05 out-name    pic
           X(25).
           05 out-addr          pic X(25).
           05 out-zip           pic 9(10).
           05 out-salary        pic 9(7)V99.

         SD member-
         sort. 01 SORT-
         DATA.
           05 sort-sequence-no PIC
           9(7). 05 sort-filler      PIC
           X(69).
        *
         WORKING-STORAGE SECTION.
         77 infile-stat  pic
         xx. 77 outfile-stat
         pic xx.

         PROCEDURE
         DIVISION.
         PRODUCT-LIST-
         SORT.
            SORT member-sort
            ON DESCENDING KEY sort-sequence-no
            USING  lab2fil
         GIVING
sortout.
```

6- Compare the results: **Lab2b.bat**

```
@echo off
now > begin2b.txt
cobcrun lab2
now >>
begin2b.txt type
begin2b.txt
```

2.3 The CitSORT Examples

For more details about the function of these examples, refer to the appropriate section of the documentation.

EXPORT a VSAM indexed file to a SEQUENTIAL file

```
>CITSORT COPY USE INDEXDFILE.DAT ORG IX RECORD F 85 KEY (1,5,P,10,3,A)
         GIVE FLATFILE.DAT ORG SQ
```

IMPORT a SEQUENTIAL file to a VSAM indexed file

```
>CITSORT COPY USE GIVE FLATFILE.DAT RECORD F 85 ORG SQ
         GIVE INDEXDFILE.DAT ORG IX KEY (1,5,P,10,3,A)
```

Add an index to a VSAM indexed file

```
>CITSORT COPY USE INDEXDFILE.DAT ORG IX RECORD F 85 KEY (1,5,P,10,3,A)
         GIVE INDEXDFILE2.DAT ORG IX KEY (1,5,P,10,3,A, 20,7,AD)
```

SIGN-EBCDIC- Placement of the SIGN-EBCDIC clause

```
>CITSORT SIGN-EBCIDIC USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D)
GIVE PRES2.TXT ORG LS
```

USE Statement, Fixed-Length RECORD clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
ORG LS
```

USE Statement, Variable-Length RECORD clause

```
>CITSORT USE PRESIDENTS2.TXT RECORD V 12 24 ORG LS SORT FIELDS (1,2,NU,A) GIVE
PRES3.TXT ORG LS
```

USE Statement, ORGANIZATION Sequential File

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 ORG SQ SORT FIELDS (1,2,NU,D) GIVE
PRES2.TXT ORG LS
```

USE Statement, ORGANIZATION Relative File

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 ORG RL SORT FIELDS (1,2,NU,D) GIVE
PRES2.TXT ORG LS
```

USE Statement, No ORGANIZATION Clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
ORG LS
```

INREC Statement, Reduce the input record to 40 bytes

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (1,40) SORT FIELDS
(1,2,NU,D) GIVE PRES2.TXT ORG LS
```

SORT Fields, Sort on Descending, Ascending Order

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
ORG LS
```

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,A) GIVE PRES2.TXT
ORG LS
```

SORT Fields, Usage of the Format Clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,A) FORMAT=NU GIVE
PRES2.TXT ORG LS
```

SORT Fields, Sort on Multiple Keys

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (43,21,CH.A,1,2,CH,A) GIVE
PRES2.TXT ORG LS
```

SORT Fields, Usage of the Format Clause, Multiple Keys

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (43,21,A,3,22,A) FORMAT=CH
GIVE PRES2.TXT ORG LS
```

SUM, Summing on a numeric field

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (43,21,41,2) SORT FIELDS
(1,21,CH,A) SUM FIELDS=(22,2,NU) GIVE PRES2.TXT ORG LS
```

SUM, Eliminating records with a duplicate sort key

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (3,21,CH,A) GIVE PRES2.TXT
ORG LS SUM FIELDS=(NONE)
```

INCLUDE Statement, Character comparion

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) INCLUDE
COND=(3,4,CH,EQ,'JOHN')GIVE PRES2.TXT ORG LS
```

INCLUDE Statement, Numeric comparison

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (1,40) SORT FIELDS
(1,2,NU,D) INCLUDE COND=(1,2,NU,EQ,10)GIVE PRES2.TXT ORG LS
```

INCLUDE Statement, Data expressed as hexadecimal

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (1,40) SORT FIELDS
(1,2,NU,D) INCLUDE COND=(1,2,NU,EQ,X'3130')GIVE PRES2.TXT ORG LS
```

INCLUDE Statement, Use of Logical AND operator

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) INCLUDE
COND=(1,2,NU,GT,01,AND,1,2,NU,LT,10) GIVE PRES2.TXT ORG LS
```

INCLUDE Statement, Use of Logical OR operator

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) INCLUDE
COND=(1,2,NU,EQ,01,OR,1,2,NU,EQ,10)GIVE PRES2.TXT ORG LS
```

INCLUDE Statement, Use of Substring operator

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) INCLUDE
COND=(3,21,SS,EQ,'JOHN')GIVE PRES2.TXT ORG LS
```

OMIT Statement, Character Comparison

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) OMIT
COND=(3,4,CH,EQ,'JOHN')GIVE PRES2.TXT ORG LS
```

OMIT Statement, Numeric Comparison

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (1,40) SORT FIELDS
(1,2,NU,D) OMIT COND=(1,2,NU,EQ,10)GIVE PRES2.TXT ORG LS
```

OMIT Statement, Data expressed as hexadecimal

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 INREC FIELDS (1,40) SORT FIELDS
(1,2,NU,D) OMIT COND=(1,2,NU,EQ,X'3130')GIVE PRES2.TXT ORG LS
```

OMIT Statement, Usage of Logical AND operator

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) OMIT
COND=(1,2,NU,GT,01,AND,1,2,NU,LT,10)GIVE PRES2.TXT ORG LS
```

OMIT Statement, Usage of Logical OR operator

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) OMIT
COND=(1,2,NU,EQ,01,OR,1,2,NU,EQ,10) GIVE PRES2.TXT ORG LS
```

OMIT Statement, Use of Substring operator

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) OMIT
COND=(3,21,SS,EQ,'JOHN')GIVE PRES2.TXT ORG LS
```

GIVE Statement, Fixed-Length RECORD clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
RECORD F 85 ORG LS
```

GIVE Statement, Organization Line Sequential

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
ORG LS
```

OUTFIL, Usage of the STARTREC Clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,D) FORMAT=NU GIVE
PRES2.TXT ORG LS OUTFIL STARTREC 40
```

OUTFIL, Usage of the ENDREC Clause

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,D) FORMAT=NU GIVE
PRES2.TXT ORG LS OUTFIL ENDREC 10
```

OUTFIL, Multiple output files, Multiple OUTRECs

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE OUT1.TXT
OUTFIL OUTREC=(1,2,' ',3,22,' ',43,21) GIVE OUT2.TXT ORG LS OUTFIL
OUTREC=(3,21,' ',25,4,'-',33,4)
```

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE OUT1.TXT
ORG LS OUTFIL STARTREC=40 OUTREC=1,2,' `,3,22,' `, 43,21) GIVE OUT2.TXT ORG LS
OUTFIL ENDREC=5 OUTREC=(3,21,' `, 25,4,'-`,33,4)
```

OUTREC Statement, Reformatting an output record

```
> CITSORT USE PRESIDENTS.DAT ORG SQ RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE
PRES3.TXT ORG LS OUTREC FIELDS=('PRESIDENT # ',1,2,' ',3,21)
```

OUTREC Statement, Reformatting output record, converting data with TO=

```
> CITSORT USE PRESIDENTS.DAT ORG SQ RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE
PRES3.TXT ORG LS OUTREC FIELDS=(1,2,NU,TO=FS,' ',3,21)
```

OUTFIL, OUTREC using CHANGE/NOMATCH

```
>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE PRES2.TXT
ORG LS OUTFIL OUTREC=(1,2,3,1,CHANGE=(1,'B','W',NOMATCH=(3,1)),4,36)
```

CitSORT MERGE functionality

```
>CITSORT MERGE USE PRESMERGE1.TXT RECORD F 85 ORG LS USE PRESMERGE2.TXT
FIELDS=(1,2,NU,A) GIVE ALLPRES.TXT ORG LS
```

CitSORT COPY functionality

```
>CITSORT FIELDS=COPY USE PRESIDENTS.DAT RECORD F 85 GIVE TESTOUT.DAT
```

Or

```
>CITSORT COPY USE PRESIDENTS.DAT RECORD F 85 GIVE TESTOUT.DAT
```

CitSORT TAKE [command-file]

```
>CITSORT TAKE PRESIDENT-PARAMS.TXT
```

2.4 CitSORT Environment Variables

CITSORT_LS_DOS

The environment variable CITSORT_LS_DOS forces record terminators for all line sequential files to be CR/LF. For existing installations, where CitSORT is being used in a DOS environment, and where scripts use an **org ls** notation, this would be required when updating to version 3.4.18, as the behavior of CITSORT has changed, and by default, the **org ls** notation causes record terminators of LF to be created.

CITSORT_LS_FIXED=V

The CITSORT_LS_FIXED environment variable, when set to « V », causes all LINE SEQUENTIAL files to assume the RECORD V (Variable Length) characteristic when created by CitSORT.

www.cobol-it.com

July, 2018