Legacy Sort Migrations: Switching to CoSort

Third-Party Sort Replacement And Conversion Tool Examples

IRI, The CoSort Company

MIS White Paper Series, Vol. 3

Table of Contents

Introduction

On mainframes, sorting is one of the most important, and resource-intensive, data processing operations. Off the mainframe, high-volume data sorting is required in commercial Unix and Windows data centers involved in re-hosting these applications, as well as in database reorg and loading operations, report generation, and data warehouse staging (ETL). In all of these cases, the robust, commercial-grade sort performance of the CoSort package is in high demand. To mitigate migration time as a factor in CoSort adoptions, IRI has developed a suite of tools to leverage your current metadata or sort function interfaces.

Mainframe Sort Migration

Mainframe sort products include DF-SORT from IBM, CA-SORT from Computer Associates, and SyncSort, from SyncSort, Inc. CoSort was the world's first commercial sort product for Unix and Windows, and has been instrumental in mainframe sort migrations since 1985. CoSort includes MVS and JCL sort parm conversion utilities, and has collaborated with every major Unix hardware manufacturer, plus many migration service providers (like Blue Phoenix, Clerity, EDS, Sungard, and Tetrad), to move mainframe sort users into open systems successfully.

"The Florida Community College Software Consortium has developed an ERP system that is supported on both the VSE and MVS platforms. To expand Consortium options for platform implementation to include Linux, solutions for utility packages, sure as sort, were required. CoSort was the perfect fit; the product was able to convert both VSE and MVS JCL with no modifications. We found CoSort to be the most cost effective, high performance solution to meet our requirements for replacing our mainframe sort products. While the migration to Linux had many challenges, CoSort prevented the sort from being one of those challenges."

Third-Party Sort Function Replacements

CoSort's brand of high-speed sorting can also directly improve the performance of many popular applications without the need for re-coding. To effect this, IRI worked with the following companies to create drop-in sort libraries or interfaces. These plug 'n' play sort facilities are now available in CoSort:

- ACUCOBOL-GT
- Clerity MBM and MTP (SortCL)
- DB2 (UDB) Loader
- Micro Focus COBOL
- SAS (Unix)
- Software AG Natural
- Unix (/bin/sort)
- IBM DataStage
- Informatica PowerCenter

In addition to sort function migration, CoSort also performs a variety of data and file format conversions.

Parameter Conversion Overview

The conversion tools provided by CoSort enable you to begin its flexible and powerful Sort Control Language (*SortCL*) program, without rewriting your existing legacy sort scripts, or creating new ones. The following diagram shows how CoSort's included tools are used in various migration scenarios:

This document describes how to migrate from the syntax of legacy sorts to CoSort's advanced 4GL for data definition (DDL) and manipulation (DML), SortCL. This conversion process can usually be performed with minimal effort by using the appropriate CoSort package utilities:

- *mvs2scl* Converts MVS JCL sort parms to CoSort SortCL job specification files
- *vse2scl* Converts VSE JCL sort parms to CoSort SortCL job specification files

In cases where CoSort tools cannot provide 100% effective conversion or replacement, CoSort engineers can usually 'fill in the gaps' free of charge. Remaining scripts can be sent to your IRI agent for manual conversion, or to request IRI's modification of a conversion tool to meet a new circumstance.

In addition to the *2scl tools above, CoSort also includes these metadata conversion utilities:

- *cob2ddf* Produces SortCL data definitions from Micro Focus COBOL copybooks
- *elf2ddf* Produces SortCL data definitions from W3C extended web log headers
- *csv2ddf* Produces SortCL data definitions from comma-separated value headers
- *ctl2ddf* Produces SortCL data definitions from SQL*Loader control files
- *ldif2ddf* Produces SortCL data definitions from LDIF files
- *xmlf2ddf* Produces SortCL data definitions from XML files

For many other applications, DDF is also a 'spoke' supported by the Meta Integration Model Bridge.

COBOL Copybook & Sort Conversions

Many of the data sources and file layouts processed on mainframes originate from COBOL applications, and are described in data dictionaries, or copybooks. For sort/merge applications being directly rehosted through CoSort on open systems, a command-line utility called '*cob2ddf*' (COBOL-to-SortCL data definition file) can immediately convert copybook layouts into the equivalent data layouts for SortCL jobs. These saved layouts are metadata repositories (or views) that can be used for repeated reference within – or direct insertion into – *SortCL* job specification files (.scl program scripts).

For a standalone conversion example, consider the following Micro Focus COBOL data dictionary that has been moved to Unix as an ASCII text file called **testl.cob**:

```
* THIS IS A SAMPLE COBOL COPYBOOK
01  EMPLOYEE-RECORD.
05  EMPLOYEE-ID.
10  RECORD-TYPE PIC A(1).
10  EMPLOYEE-CODE PIC 9(6).
05  EMPLOYEE-NAME.
10  NAME-FIRST PIC X(9).
10  NAME-LAST PIC X(15).
10  NAME-MIDDLE-I PIC A.
05  EMPLOYEE-ADDRESS.
10  STREET-ADDR PIC X(20).
10  CITY PIC A(10).
```

CoSort's *cob2ddf* utility can then be run on the command line as follows:

```
$  cob2ddf      testl.cob        test1.ddf
```

This produces the following data definition file:

```
$ cat test1.ddf

/FIELD=(RECORD_TYPE, POSITION=1, SIZE=1, CHAR)
/FIELD=(EMPLOYEE_CODE, POSITION=2, SIZE=6, NUMERIC)
/FIELD=(NAME_FIRST, POSITION=8, SIZE=9, ASCII)
/FIELD=(NAME_LAST, POSITION=17, SIZE=15, ASCII)
/FIELD=(NAME_MIDDLE_I, POSITION=32, SIZE=1, CHAR)
/FIELD=(STREET_ADDR, POSITION=33, SIZE=20, ASCII)
```

The above file can then be referenced from within any applicable SortCL job specification (.scl) using the /SPECIFICATION command, for example:

```
# CoSort (SortCL) Job Specification File
# testl.scl created 3/20/09 by dwh, MIS
/INFILE=employee
/SPEC=testl.ddf
/KEY=EMPLOYEE_CODE
/OUTFILE=employees.srt
/STATISTICS
```


In this case, the EMPLOYEE_CODE field is used as the key. Any SortCL field layout or command statement can be centrally stored in .ddf files and referenced repeatedly.

In this next case, *cob2ddf* ignores several irrelevant statements in the original copybook:

```
01 OSTRINGS-CLASS.
   02 OSTRINGS-IHANDLE POINTER.
   02 OSTRINGS-CREATE-METHOD.
      03 OSTRINGS-CREATE-RETV POINTER.
      03 NAME-IN PIC X(1024)
   02 MSTRING-IOM-METHOD.
      03 ARG1A-IN PIC X(1024).
      03 ARG1B-IN PIC X(1024).
      03 ARG1C-IN PIC X(1024).
      03 ARG2A-OUT POINTER.
      03 ARG2B-OUT POINTER.
      03 ARG2C-OUT POINTER.
   02 MSTRING-IOR-METHOD.
      03 MSTRING-IOR-RETV POINTER.
      03 ARG1-IN PIC X(1024).
      03 ARG2-OUT POINTER.
      03 INDEX-IN PIC S9(8) COMP.
   02 MSTRING-IR-METHOD.
      03 MSTRING-IR-RETV POINTER.
      03 ARG1-IN PIC X(1024).
   02 MSTRING-O-METHOD.
      03 ARG1-OUT POINTER.
      03 INDEX-IN PIC S9(8) COMP.
```

When run on the command line, *cob2ddf's* error messages, such as the one below, are displayed:

```
$  cob2ddf   test2.cob   test2.ddf

Test2.con:02: error: POINTER usage unsupported
```


However, the proper *SortCL* field layouts are still produced:

```
$ cat test2.ddf

/FIELD=(NAME_IN, POSITION=1, SIZE=1024, ASCII)
/FIELD=(ARG1A_IN, POSITION=1025, SIZE=1024, ASCII)
/FIELD=(ARG1B_IN, POSITION=2049, SIZE=1024, ASCII)
/FIELD=(ARG1C_IN, POSITION=3073, SIZE=1024, ASCII)
/FIELD=(ARG1_IN, POSITION=4097, SIZE=1024, ASCII)
/FIELD=(INDEX_IN, POSITION=5121, SIZE=8, MF_COMP)
/FIELD=(ARG1_IN, POSITION=5129, SIZE=1024, ASCII)
/FIELD=(INDEX_IN, POSITION=6153, SIZE=8, MF_COMP)
```

Again, these layouts can be directly inserted into a **.scl** script's /INFILE, /INREC or /OUTFILE sections, or referenced for convenience with the statement: `/SPECIFICATION=test2.ddf`. It does not matter if the data definition file contains field layouts that are not used by a given application. Moreover, extra field layouts in a metadata repository that extend the view can be declared within any *SortCL* job script.

Unrelated to metadata, but for the information of COBOL users, IRI has a drop-in replacement for the ACUCOBOL-GT sort verb on Unix using the CoSort engine. IRI has also enabled ACUCOBOL Vision index file support in CoSort's *SortCL* (for collation and conversion/reformatting). CoSort packages also include drop-in replacements for the sort verb within Micro Focus Workbench 4.0 for Windows and 4.1 for Unix, Net Express 3.1 and 4.x for Windows, and Server Express 4-5 for Unix.

Finally, COBOL users can also call *SortCL* or any of CoSort's API libraries from within their programs.

MVS JCL Sort Re-hosting

For sort/merge users migrating JCL sort steps from an IBM z/OS mainframe environment to open systems, conversion to CoSort's sort control language (SortCL) program scripts on Unix or Windows is a simple two-step process:

1.) Move the JCL job stream(s) containing sort cards to the open systems platform.

2.) Run the JCL through CoSort's MVS-to-SortCL (*mvs2scl*) conversion program.

The *mvs2scl* program scans the mainframe sort parms to produce an executable *SortCL* script. If necessary, input (and output) file metadata descriptions can be translated from COBOL copybook layouts to *SortCL*-supported .ddf files using the *cob2ddf* utility discussed in the previous section.

CoSort's *mvs2scl* process any MVS mainframe JCL sort, and will process the basic JCL file and SYSIN, $SORTPARM, and DFSPARM-associated files, if they are present. *mvs2scl* handles multiple job steps without additional processing, and can generate either ASCII or EBCDIC collating sequence sorts based on a command line flag (option).

In addition, *mvs2scl* defines the mapping between MVS dataset names and Unix filenames, and can be set to retain the original JCL as comments in the shell file, or generate only the equivalent *SortCL* specifications. mvs2scl can leave non-sort-related JCL in the shell script if further processes require it.

For a standalone conversion example, consider the following MVS JCL script, called mvs1.jcl:


```
//JUBQ520   JOB (B09999),GI,
//          MSGCLASS=S
//          COND=(8,LT),
//          REGION=0M
//JUBQ525   EXEC PGM=SORT
//SYSPRINT  DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//SORTWK01  DD UNIT=DNAKS,SPACE=(CYL,(10,150))
//SORTWK02  DD UNIT=DNAKS,SPACE=(CYL,(10,150))
//SORTWK03  DD UNIT=DNAKS,SPACE=(CYL,(10,150))
//SORTWK04  DD UNIT=DNAKS,SPACE=(CYL,(10,150))
//SORTWK05  DD UNIT=DNAKS,SPACE=(CYL,(10,150))
//SORTWK06  DD UNIT=DNAKS,SPACE=(CYL,(10,150))
//SORTIN    DD DSN=QV.F58700.IN.JUBQ510.JUFETATS,
//             DCB=(LRECL=1504,RECFM=VB),DISP=(OLD,KEEP)
//SORTOUT   DD DSN=QV.58700.IN.JUBQ520.JUFETATS,
//             UNIT=DNAKS,DISP=(,CATLF,DELETE),
//             SPACE=(CYL,(9,150),RLSE),
//             LRECL=1504,RECFM=VB
//SYSIN     DD *
SORT FIELDS=(5,180,BI, A)
INCLUDE COND=((40,15,CH,EQ,C'720141000700150'),OR,
   (40,15,CH,EQ,C'720141000702091'),OR,
   (40,15,CH,EQ,C'720101000942692'),OR,
   (40,15,CH,EQ,C'720111000800297'))
//
/*
```

To perform the conversion that creates the SortCL script, enter: **`$ mvs2scl mvsl.jcl mvsl.scl`**
This produces the following *SortCL* job specification (text) file:

```
$cat mvs1.scl

/INFILE=(QV.F58700.IN.JUBQ510.JUFETATS)
  /LENGTH=1504
  /FIELD=(field_0, POSITION=5, SIZE=180, MF_COMP)
  /FIELD=(field_1, POSITION=40, SIZE=15, EBCDIC)
  /CONDITION=(cond_0, TEST=(field_1 =="720141000700159" OR \
  field_1 == "720141000702091" OR \
  field_1 == "720101000942692" OR \
  field_1 == "720111000800297" OR \
  /INCLUDE=(CONDITION=cond_0)
/KEY=(field_0, ASCENDING)
/OUTFILE=(QV.F58700.IN.JUBQ520.JUFETATS
```

The command line statement used to produce the same output file is: `$sortcl /spec=mvsl.scl`

Note that the resulting *SortCL* field statements have field names (field_0, field_1, etc.) that are associated with the file's column positions and lengths. Subsequent references to these same fields (such as in key, condition, and summary statements) allow *SortCL* users to easily manipulate and remap their data. Accordingly, these field names should be changed to reflect their symbolic meaning within the records. Any statement in the original MVS script that cannot be translated will be ignored, such as references to physical locations (i.e disk storage and cylinders). Also, many MVS operating system references are not applicable in the Unix or Windows environment.

Another feature of CoSort's *mvs2scl* utility is its ability to preserve the original JCL and intersperse it with the required *SortCL* output. For example, consider the following JCL job stream, **mvs2.jcl**:

```
//JUBU740   JOB (B09999),GI,
//          MSGCLASS=S
//          COND=(8,LT),
//          REGION=0M
//*****************************************************************
//JUBU740A EXEC PGM=RZ271300,PARM='SCAN=TIOT',
// ACCT=(B08888)
//SYSPRINT  DD SYSOUT=*
//DP1001A   DD DSN=C.H93010.VARH,DISP=(OLD,KEEP),UNIT=SYSDA
//*----------------------------------------------------------------*
//IF JUBU740A.RC=0 ! JUBU740A.RUN=FALSE THEN
//*****************************************************************
//*         SORTING SECTION                                       *
//*****************************************************************
//JUBU740B EXEC PGM=SORT,
// ACCT=(B09999)
//SYSOUT    DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SORTWK01  DD UNIT=SYSDA.SPACE=(CYL,5)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,5)
//SORTIN    DD DSN=K.H93010.VARH(0),     /*FROM JUBU730
//             DISP=(OLD,KEEP)
//SORTOUT   DD DSN=C.H93010.VARH,DISP=(OLD,KEEP)
//SYSIN     DD *
 SORT FIELDS=(1,19,A),FORMAT=BI
//*
//*----------------------------------------------------------------*
//ENDIF
```


By running **$mvs2scl –p mvs2.jcl mvs.scl**, the **mvs2.scl** result contains:

```
//JUBU740   JOB (B09999),GI,
//          MSGCLASS=S
//          COND=(8,LT),
//          REGION=0M
//*****************************************************************
//JUBU740A EXEC PGM=RZ271300,PARM='SCAN=TIOT',
// ACCT=(B08888)
//SYSPRINT  DD SYSOUT=*
//DP1001A   DD DSN=C.H93010.VARH,DISP=(OLD,KEEP),UNIT=SYSDA
//*----------------------------------------------------------------*
//IF JUBU740A.RC=0 ! JUBU740A.RUN=FALSE THEN
//*****************************************************************
//*         SORTING SECTION                                        *
//*****************************************************************
//JUBU740B EXEC PGM=SORT,
// ACCT=(B09999)
//SYSOUT    DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SORTWK01  DD UNIT=SYSDA.SPACE=(CYL,5)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,5)
//SORTIN    DD DSN=K.H93010.VARH(0),     /*FROM JUBU730
//             DISP=(OLD,KEEP)
//SORTOUT   DD DSN=C.H93010.VARH,DISP=(OLD,KEEP)
//SYSIN     DD *
/INFILE=(K.H93010.VARH)
     /FIELD=(field_0, POSITION=1, SIZE=19, MF_COMP)
     /KEY=(field_0, ASCENDING)
/OUTFILE=(C.H93010.VARH)
 SORT FIELDS=(1,19,A),FORMAT=BI
//*
//*----------------------------------------------------------------*
//ENDIF
```


Modified or additional *SortCL* data manipulation statements can be added into the script, or on the command line at runtime. In addition to the legacy functions of sorting, selecting, and summing, CoSort's SortCL program has many data manipulation features that exceed MVS sort products:

- multiple input and output files, each in their own formats
- data types and file formats translated in the same pass
- static and running summaries with totals, counts, averages, minimum and maximum values
- mathematical and trigonometric functions across the records
- break conditions for intra- and inter-record events
- mixed INCLUDE-OMIT statements
- support for C, FORTRAN, MF, and RM COBOL data, as well as timestamps
- Micro Focus Variable Length, ELF, CSV, UNIVBF, Vision, LDIF and other file processing
- user exits for input, compare, and output routines for customized job criteria
- support for central file/record data definition sharing (metadata repositories)
- cross-table join (matching) and lookup functionality for flat files
- auditable, field-level de-identification, encryption, and other data masking functions
- interface to cross-platform job design and remote execution GUI

VSE JCL Sort Re-hosting

For sort/merge users migrating JCL sort steps from an IBM VSE mainframe environment to open systems, conversion to CoSort's sort control language (*SortCL*) program scripts on Unix or Windows is a simple two-step process:

1.) Move the JCL job stream(s) containing sort cards to the open systems platform.

2.) Run the JCL through CoSort's VSE-to-SortCL (vse2*scl*) conversion program.

The vse2scl program scans the mainframe sort parms to produce an executable *SortCL* script. If necessary, input (and output) file metadata descriptions can be translated from COBOL copybook layouts to *SortCL*-supported .ddf files using the *cob2ddf* utility discussed previously.

vse2scl handles multiple job steps without additional processing, and can generate either ASCII or EBCDIC collating sequence sorts based on a command line flag (option). In addition, *vse2scl* defines the mapping between VSE dataset names and Unix filenames, and can be set to retain the original JCL as comments in the shell file, or generate only the equivalent *SortCL*. *vse2scl* can leave non-sort-related JCL in the shell script if further processes require it.

For a standalone conversion example, consider the following VSE JCL script, called **vse1.jcl**:

```
// DLBL  SYS011,'CAP.P.AUTO.PAC',,VSAM,DISP=NEW,CAT=PRODCAT 06760
// DLBL  REPO260,'CAP.N.AUTO.REP0260A',,VSAM,CAT=PRODCAT
// DLBL  SORTIN1,'CAP.N.AUTO.REP0260A',,VSAM,CAT=PRODCAT
// DLBL  SORTOUT,'CAP.N.AUTO.REP0260A',,VSAM,DISP=NEW,CAT=PRODCAT
* PCAPA260 - SORT:PAC TAPE DATA FOR NN 06720
// EXEC  SORT,SIZE=256K
        SORT FIELDS=(1,2,A,29,7,A,3,26,A,36,10,A),FORMAT=CH
        RECORD TYPE=F,LENGTH=(170)
        INPFIL VSAM
        OUTFIL ESDS,REUSE
        OPTION ROUTE=LST
        END
/*
```

To perform the conversion that creates the *SortCL* script, enter:

```
$ vse2scl vsel.jcl vsel.scl
```


This produces the following *SortCL* job specification (text) file:

```
$ cat vse1.scl

/INFILE=(CAP.N.AUTO.REP0260A)
     /LENGTH=170
     /FIELD=(field_0, POSITION=1, SIZE=2, EBCDIC)
     /FIELD=(field_1, POSITION=29, SIZE=7, EBCDIC)
     /FIELD=(field_2, POSITION=3, SIZE=26, EBCDIC)
     /FIELD=(field_3, POSITION=36, SIZE=10, EBCDIC)
/KEY=(field_0, ASCENDING)
/KEY=(field_1, ASCENDING)
/KEY=(field_2, ASCENDING)
/KEY=(field_3, ASCENDING)
/OUTFILE=(CAP.N.AUTO.REP0260A)
```

The command line required to produce the same output file is simply:

```
$ sortcl /spec=vse1.scl
```

Note that the resulting *SortCL* field statements have field names (field_0, field_1, etc.) that are associated with the file's column positions and lengths. Subsequent references to these same fields (such as in key, condition, and summary statements) allow SortCL users to easily manipulate and remap their data. Accordingly, these field names should be changed to reflect their symbolic meaning within the records.

Like the *mvs2scl* utility, *vse2scl* uses the –p flag to optionally preserve the original JCL parms and intersperse them with the required SortCL output specifications.

vse2scl users can also use –e or –a to determine if data declared as CH in JCL will be typed as EBCDIC (default), or ASCII, respectively, in the resulting SortCL job specification file. For example:

To perform the conversion that creates a script with ASCII field specifications, enter:

```
$ vse2scl –a vse1.jcl vse2.scl
```

This produces the following *SortCL* job specification (text) file:

```
$ cat vse2.scl

/INFILE=(CAP.N.AUTO.REP0260A)
     /LENGTH=170
     /FIELD=(field_0, POSITION=1, SIZE=2, ASCII)
     /FIELD=(field_1, POSITION=29, SIZE=7, ASCII)
     /FIELD=(field_2, POSITION=3, SIZE=26, ASCII)
     /FIELD=(field_3, POSITION=36, SIZE=10, ASCII)
/KEY=(field_0, ASCENDING)
/KEY=(field_1, ASCENDING)
/KEY=(field_2, ASCENDING)
/KEY=(field_3, ASCENDING)
/OUTFILE=(CAP.N.AUTO.REP0260A)
```


Modified or additional *SortCL* data manipulation statements can be added into the script or on the command line at runtime. In addition to the legacy functions of sorting, selecting, and summing, CoSort's SortCL program has many data manipulation features that exceed VSE sort products:

- multiple input and output files, each in their own formats
- data types and file formats translated in the same pass
- static and running summaries with totals, counts, averages, minimum and maximum values
- mathematical and trigonometric functions across the records
- break conditions for intra-record and inter-record events
- mixed INCLUDE-OMIT statements
- support for C, FORTRAN, MF and RM COBOL data, as well as timestamps
- Micro Focus Variable Length, ELF CSV, UNIVBF, Vision, LDIF and other file processing
- user exits for input, compare, and output routines for customized job criteria
- support for central file/record data definition sharing (metadata repositories)
- cross-table join (matching) and lookup functionality for flat files
- auditable, field-level de-identification, encryption, and other data masking functions
- interface to a cross-platform job design and remote execution GUI

Replacing the Unix/bin/sort

The /bin/sort utility provided with the Unix operating system is designed for ordering small collections of alphanumeric data. It does not work well when the size of the data increases beyond available memory. For those with an investment in, or familiarity with, existing system sort commands, IRI provides a drop-in /bin/sort replacement tool that improves high volume sort performance through the **CoSort** engine.

Unix users are provided with the new **/bin/sort**, and Windows users get a **unixsort.exe** program.

Upon installation, the object file can be moved into a system directory to provide sort services with the same syntax as the original sort verb, but at much higher performance levels.

An example of CoSort's Unix sort replacement follows, using this mock input file, **chicago**:

```
5180 On Top 15.95 Harper-Row
3391 Married Young 24.95 Prentice-Hall
8835 Beginnings 8.50 Prentice-Hall
2272 Still There 13.05 Dell
1139 Greater Than 34.75 Valley Kill
3928 Not On Call 9.99 Harper-Row
4877 Going Nowhere 17.95 Valley Kill
```

We first define the way to specify fields for the sort key. White space denotes the end of a field unless a field separator character is defined.

To sort **chicago** starting with the second character of the first field, use the following command:

```
Sort –k 1.2 chicago
```

This results in the following output:

```
1139 Greater Than 34.75 Valley Kill
5180 On Top 15.95 Harper-Row
2272 Still There 13.05 Dell
3391 Married Young 24.95 Prentice-Hall
8835 Beginning 8.50 Prentice-Hall
4877 Going Nowhere 17.95 Valley Kill
3928 Not On Call 9.99 Harper-Row
```


CoSort's Unix sort replacement supports these command-line flags:

```
-c          -d
-m          -f
-u          -i
-o          -M
-T          -b
-n          -t
-r          -z
```

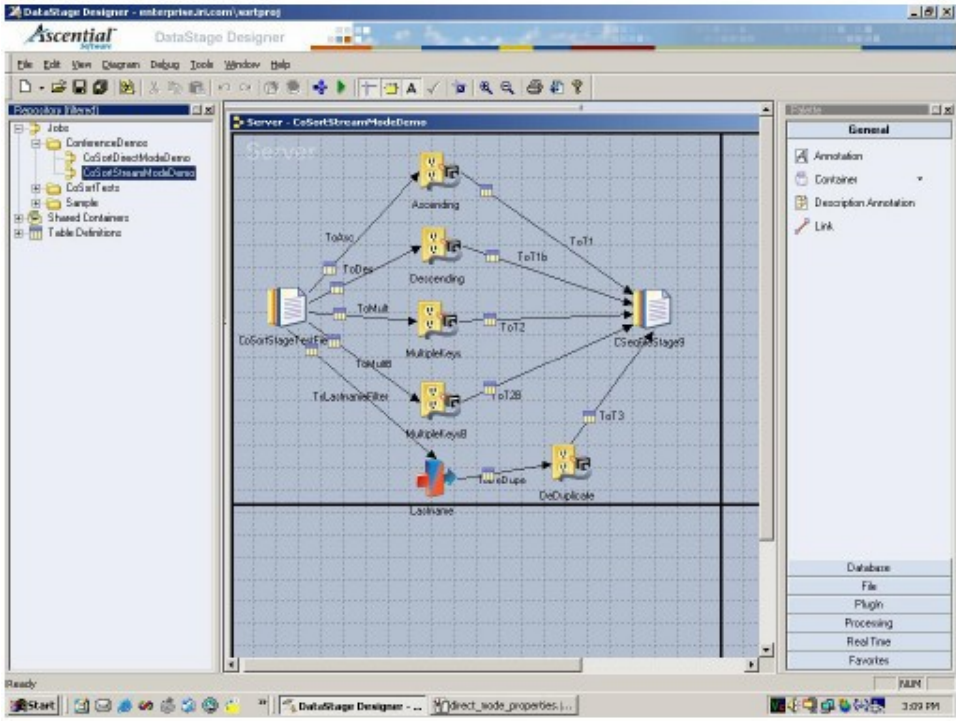
-y and –Kmem are supported indirectly, via values in the CoSort Resource Control (cosortrc) file on Unix and Linux or the Windows registry.

Sort Stage PlugIn for DataStage

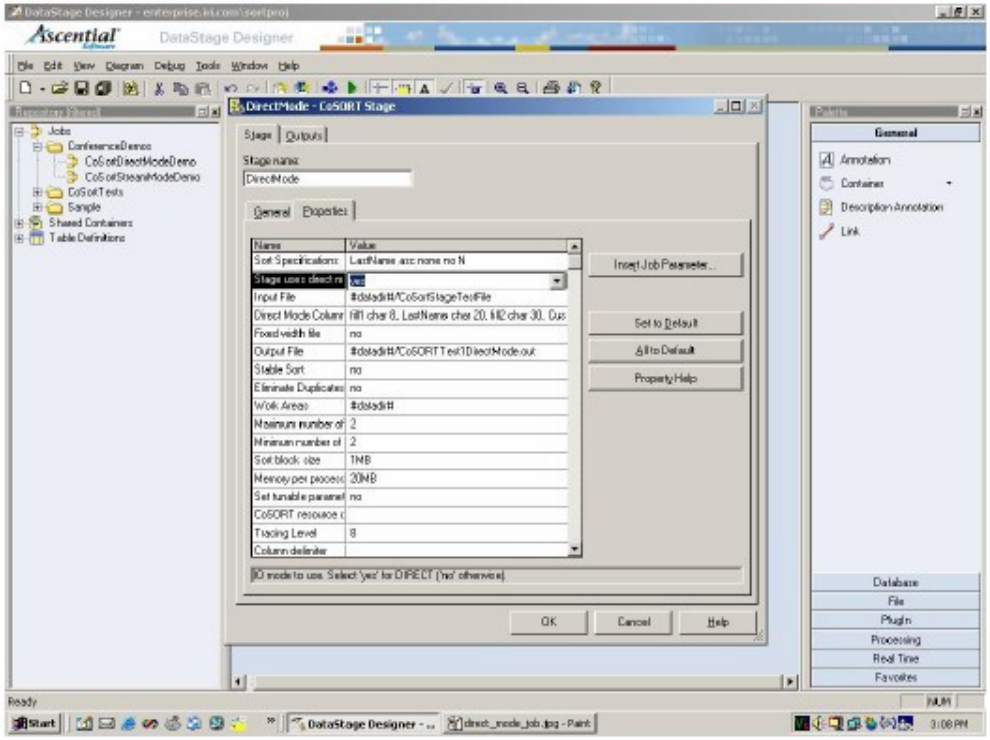
The CoSort Sort Stage Plug-In for DataStage was initially developed with Informix software in 1999, and released in 2000 with DataStage 4.5. Now available from IBM WebSphere DataStage Server Edition 7.5.2, the unique CoSort stage can speed sorting operations dramatically with out major workflow impacts. In addition to sort performance, subsequent join, aggregation, and load runtimes should also benefit.

File Size	Rows	Native DS Sort Stage	CoSort PlugIn	CoSort SortCL
63MB	1,000,000	6 minutes	3 minutes	15 seconds
157MB	2,500,000	22 minutes	7 minutes	42 seconds
630MB	10,000,000	2 hours, 47 minutes	37 minutes	3 minutes

Platform: 2-CPU Sun 280R with only 512MB RAM



CoSort Sort Stage Stream Mode Operation



CoSort
Sort Stage
Direct Mode
Properties

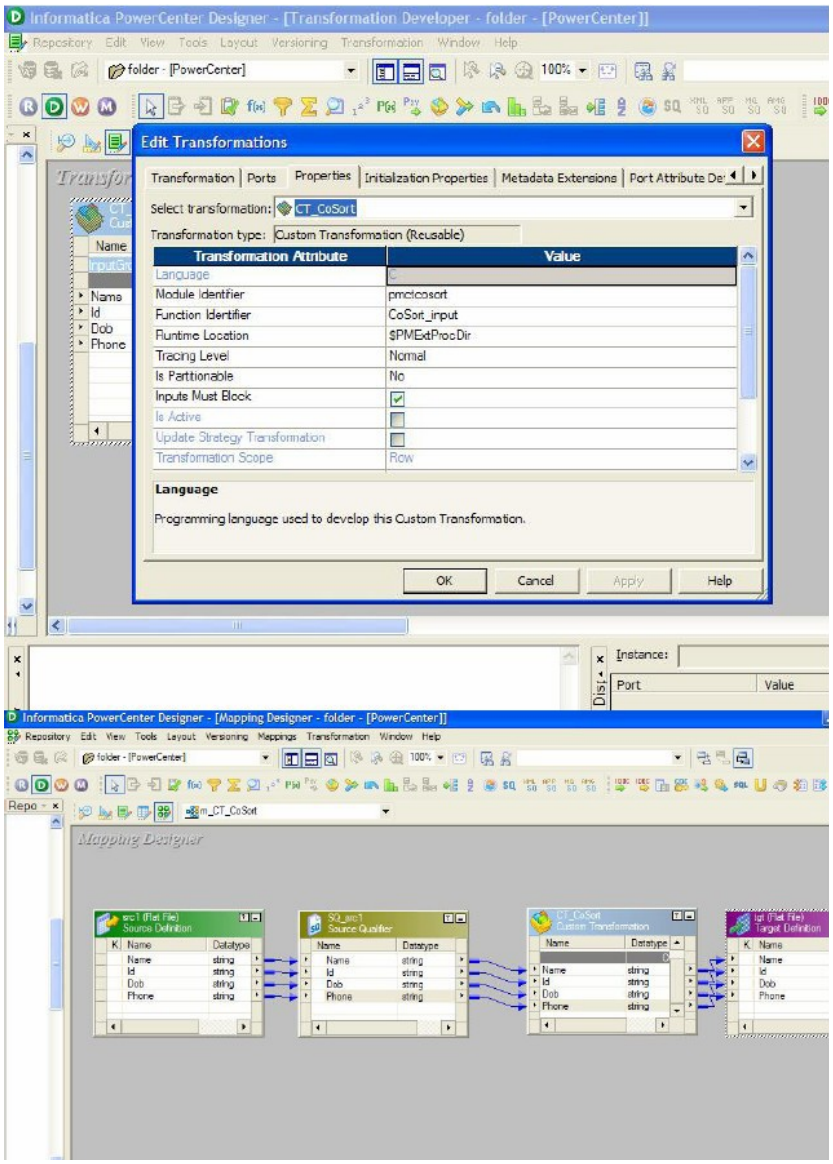
CoSort's *SortCL* program can also run on flat files outside of DataStage to improve the performance of sorts, joins, aggregations and loads, and combine these transforms with field-level data protections (like encryption) and produce custom, formatted reports.

Custom Transform for Informatica Sorter Tx

The CoSort Sorter Tx replacement for Informatica was first available in PowerCenter v6.2, as an Advanced External Procedure. Now available for PowerCenter 8.x, the CoSort Custom Transform is still faster than the default sort transformation.

Input File Size	IPC 7 Sworter Tx	CoSort v8 AEP	CoSort SortCL
26.7MB	8 seconds	3 seconds	1 second
267MB	1 minute, 48 seconds	16 seconds	7 seconds
2.67MB	20 minutes, 35 seconds	2 minutes, 1 second	2 minutes, 48 seconds

Platform: IBM p650, 4-CPUs, 32MB of RAM used



CoSort Custom Sort Transform - Properties

CoSort Custom Sort Transform - Mapping

CoSort's *SortCL* program can also run on flat files outside of PowerCenter to improve the performance of sorts, joins, aggregations and loads, and combine these transforms with field-level data protections (like encryption) and produce custom, formatted reports.

Additional Migration Technologies

SAS Sort Replacement

One of the principal legacy data processing tools on mainframes today is the SAS system. SAS has written plug-in PROC sort 'appendages' to the shared CoSort library routines for several Unix flavors. SAS System 8 and System 9 can make calls to dynamic libcosort routines to improve sort performance. Making CoSort available as the default sort within SAS requires installing CoSort and defining a few links on the command line.

Software AG Natural Sort Replacement

Another data processing tool migrated from mainframe Adabas environments is Software AG's Natural 4GL. IRI has also produced a drop-in replacement for the sort verb within Natural on Unix to vastly improve sort performance. Making CoSort available as the default sort within Natural simply requires installing CoSort and defining a few links on the command line.

CoSort Load Accelerator for DB2

When IBM migrated its mainframe database to Unix systems, high-volume load performance became a perceived bottleneck at CoSort sites. In conjunction with IBM Canada, IRI created the "CoSort Load Accelerator for DB2" which seamlessly replaces the sort verb within the loader to reduce load times.

CoSort or NextForm – Legacy Data Migration

CoSort users migrating from the mainframe not only need to convert JCL sort parms and copybook metadata to SortCL, but they often need to convert their data as well, to prepare it for open systems processing – or vice versa (convert back into mainframe format). Using the same field and file definition syntax of SortCL, either CoSort, or IRI's new standalone NextForm tool, can be used to cost-effectively convert file formats, record layouts, and data types – and all in a single pass.

Both SortCL and NextForm can automatically change file and field formats between mainframe and workstation platforms without losing data or changing existing record layouts (unless required).

Supported file formats include:

CSV	LDIF	Micro FocusVariable Length	Micro Focus ISAM	Line Sequential	Record Sequential
Text	Blocked	Variable Sequential	Vision	VSAM	XML

Very large file conversions are possible with the tools. Reformatting can also consist of changing field layouts from fixed to variable (or vice versa), moving fields around, displaying only certain fields on

output, and/or creating create multiple output files (and formats) from one input format.

Supported data types for conversion are in these groups:

C (ASCII) /Numeric	RM COBOL	MF COBOL
EBCDIC RM COBOL	EBCDIC MF COBOL	Date / Timestamp
Single-Byte	Multi-Byte	Zoned

INNOVATIVE ROUTINES INTERNATIONAL (IRI), INC.
Suite 303, Atlantis Center
2194 Highway A1A
Melbourne, FL 32937-4932 USA
Phone 321.777.8889
Fax 321.777.8886
http://www.iri.com