CoSort® Version 10.0.1
Innovative Routines International (IRI), Inc.
June 30, 2018
NDA CONFIDENTIAL: CoSort Release Notes

This readme file contains release notes that supplement CoSort's product manual and on-line help in the IRI Workbench IDE for CoSort. You may print a copy of the release notes but you may not transmit, disclose or post their contents without the prior written consent of IRI, Inc. If you have any questions about the use or contents of these notes, please contact IRI:


CALL 1-321-777-8889, ext. 3
EMAIL info@iri.com
URL http://www.iri.com
WRITE Innovative Routines International (IRI), Inc.
2194 Highway A1A, Suite 303
Melbourne, FL 32937-4932
United States of America


The release notes are divided into these sections:



These notes describe enhancements to CoSort® for UNIX®, Linux® and Microsoft® Windows® effective since Release 9.5.3. Also note that the:

changes from 9.1.x to 9.5.x are in the version 9.5 Readme files.
changes from 8.2.x to 9.1.x are in the Version 9.1 Readme files.
changes from 8.1.x to 8.2.x are in the Version 8.2 Readme files.
changes from 7.5.x to 8.1.x are in the Version 8.1 Readme files.
changes prior to 7.5.x may be available; contact cosort@iri.com.

Most of these files are available upon request from CoSort users who are upgrading from older versions of CoSort.

For IBM® iSeries® users running Linux or OS/400® PASE, refer to the Unix instructions in this file and other CoSort documentation. See the list of currently-supported CoSort platforms under www.iri.com/products/cosort.

## (1) PRODUCT DIRECTORY ORGANIZATION

CoSort software will be installed into the default directories shown below. Both Unix and Windows users can utilize the same job scripts and performance parameters across both platforms.

The base directory where CoSort is installed must be set as the value of an environment variable named COSORT_HOME. This location is referenced by the environment variable %COSORT_HOME% on Windows, and $COSORT_HOME on other platforms. On Unix the system administrator typically installs CoSort into a base directory named cosort100/ below any path, such as /usr/local/cosort100. This location is then set as the value of $COSORT_HOME. On Windows the default installation behavior is to create a base directory at C:\IRI\CoSort100. This location is then set as the value of %COSORT_HOME%. The following list of subdirectories are relative to the base install directory, which is the value of COSORT_HOME:

| Directory | Contents |
|---|---|
| bin | executables, conversion tools |
| docs | manuals, readme file, license agreement |
| etc | tuning, license, error and log files |
| examples | paths to sample specifications |
| include | headers, defines, errors, etc. |
| lib | user exit, plug-ins, and API libraries |
| lib/modules | external libraries that auto-load at runtime |
| sets | sample files for masking and test data jobs |

IRI Workbench, Built on Eclipse™, is the Integrated Development Environment (IDE) for the CoSort Sort Control Language (SortCL) program. Its purpose is to create, import, modify, save, print, and launch SortCL job scripts. Jobs can execute locally, or on a remote host over a TCP/IP connection. IRI Workbench is available at no cost to CoSort users. IRI Workbench can be installed separately, or as part of the CoSort full install package for Windows. The default install location for IRI Workbench will vary by platform, and whether it is installed for all users, or a specific user.

**(2) SETUP PROGRAM AND LICENSING PROCEDURE**

On both Unix and Windows systems, CoSort uses similar installation and configuration programs to combine loading, licensing, and resource tuning. "cs_setup" programs are run automatically at first time installation, and can be re-run when updating the license or tuning parameters. The cs_setup program will prompt you for responses based on your current configuration and job requirements; see (3) below.

First time installers must obtain a 3-part license (activation) key from IRI to run CoSort utilities or enable API calls to CoSort libraries. The key allow you to easily conform to the terms of an evaluation or permanent license, and allow upgrades to CoSort or your hardware, usually without new shipments.

CoSort 10 for Unix and Windows uses a central license file named "cosort.lic" which contains specific a license key for each machine.  By keeping licensing information central to the machine instead of the executable, users can update specific files via downloaded patch files, without the need to reinstall, re-register, or re-key the product. Prior to version 10, license and tuning information was stored in the registry on Windows.

**(3) MULTI-THREADED PERFORMANCE AND TUNING**

To scale high-volume data processing requirements for very large data sets and data warehouses, CoSort performs sorting and related data transformations across multiple cores on systems where supported. CoSort 10 improves automatic management of system resources to maximize the efficiency of CoSort along with other applications running in a multi-tasking environment.

Both the Unix and Windows versions of CoSort help set global default system tuning values interactively during initial setup. These values are stored in a CoSort Resource Control file named "cosort.rc" in $COSORT_HOME/etc. Prior to version 10, the default tuning file on Unix and Linux platforms was named "cosortrc". This file name is still supported on all platforms in version 10, but "cosort.rc" is now preferred.

The global tuning values can be overridden by creating a local tuning override files. On Unix, or a "cosort.rc" file in the same directory as a CoSort executable on Windows).

The following resource controls are created in CoSort 10, on both Unix and Windows platforms:

| Control Name <value> | Default Value | Description |
|---|---|---|
| THREAD_MAX <count> | As licensed | max # of sorting CPUs |
| THREAD_MIN <count> | 1 | min # of sorting CPUs |
| MEMORY_MAX <AUTO,MINIMIZE,#,%> | AUTO | RAM for sort buffers |
| WORK_AREA <directories> | ./ | overflow (temp) path/s |
| BLOCKSIZE <bytes> | 1200/3584KB | size of I/O buffers |
| MINIMUM_YEAR <2-digit year> | 70 | sliding century window |
| LOG [path]<file name> | | continuous log file |
| MONITOR_LEVEL <0-9> | 1 | running message detail |
| ON_EMPTY_INPUT <option> | PROCESS_WITH_ZEROS | sortcl output displays |
| ON_WORKAREAS_FULL <option> | ABORT | pause/resume behavior |
| OUTPUT_TERMINATOR <option> | INFILE | VL record terminators |
| AUDIT [path]<file name> | | XML audit file |

During setup, it is recommended that you allow the MEMORY_MAX value to be set to AUTO. If however you will be running multiple, large sortcl jobs at the same time, set MEMORY_MAX to MINIMIZE to automatically minimize the use of RAM.

Additionally available, advanced RC settings documented in the manual include:

| | |
|---|---|
| ON_MISSING_OUTLENGTH | ON_CONVERSION_FAILURE |
| AIO | PREVENT_DOUBLE_TERM |
| AIO_BUFFERS | USE_RECORDCOUNT_API |
| AIO_RETRY_TIMEOUT | COMPRESS_WORKFILES |
| ON_EMPTY_OUTPUT | ON_EMPTY_OBJECT_VALUE |
| SUMMARY_OVERFLOW_BREAK | PRESERVE_ERROR_LOG |
| ENDIANNESS | ZIP_OUT_LEVEL |
| ON_COLLATION_FAILURE | ON_FIELD_OVERFLOW |

The following, now obsolete RC settings, are removed (not supported):

```
MEMORY_PERPROCESSOR_MIN                     OUTPUT_PROCESS
MEMORY_PERPROCESSOR_MAX                     AUTO_MEM
EXTERNAL_MAXMEM                             PERCENT_SHARED
MERGE_WORKFILES_PERTHREAD_MIN               CHECK_DISK_FREE
MERGE_WORKFILES_PERPROCESSOR_MIN            IN_ACCEL
BUFFER_IPC_MIN                              OUT_ACCEL
BUFFER_IPC_MAX                              OUTPUT_PROCESS
BUFFER_IO
```

CONSERVE_CPU has been replaced by COMPRESS_WORKFILES.
PROCESSOR_MIN is deprecated and has been replaced by THREAD_MIN
PROCESSOR_MAX is deprecated and has been replaced by THREAD_MAX

Look in Appendix D of the User Manual & Programmer's Guide for more information on setting and overriding the values, on the order of precedence for multiple resource settings, and for additionally available global resource or SortCL application specific parameters. On all platforms, you can see the settings in effect prior to an execution by executing "sortcl /rc".

Additional performance recommendations:

WORK_AREA
---
For the best sort performance on multi-CPU systems specify up to 4 directories on different available physical disks. For example, in the cosort.rc file, to specify two disks:

```
    WORK_AREA       /export/home1/sorttemp
    WORK_AREA       /export/home2/sorttemp
```

Sort work files will be read simultaneously from both. Also important, try not to specify work areas that are on the same physical disks as the input or output files.

LICENSE LIMITS
---
The values in your resource control files only apply if your system, job, and CoSort license allows them; e.g., you cannot specify THREAD_MAX 8 if your company has only acquired a license to use 4 threads. Similarly, if your CPUs are hyper-threaded, CoSort will not utilize them if the additional core threads were not licensed.

GET HELP
---
Manual system tuning can override automatic and default settings provided by IRI for your environment. Only qualified users and system administrators should modify the values in resource control files, since changes can impact critical sort and non-sort jobs. Preceding configuration file names with a dot can hide them normal directory listings. Either your local IRI representative or a CoSort support engineer (email support@iri.com) can review your resource settings and system information, and recommend changes.

**(4) PRODUCT ENHANCEMENTS (since 9.5.3)**

a) Automatic Tuning

For sort jobs of any size, a MEMORY_MAX setting of AUTO will optimize the amount of memory for each job, and if insufficient, optimizes I/O. For multiple simultaneous sorts, setting MEMORY_MAX to MINIMIZE automatically reduces the amount of memory each job allocates, improving the throughput of every job running.

b) Module Directory & Version Checking

Version checking between external libraries and the main sortcl program has been added for improved reliability. External modules are checked to verify that they were built using the same version of dependencies that the sortcl program used when built. Previously a module older or newer than the sortcl program could cause unexpected behavior. Modules built by IRI now use this feature; custom external functions can optionally use it.

c) Bug Fixes

Observed or reported conditions in 9.5.3 releases resulting in memory leaks or buffer overflows have been resolved. UTF8 & UTF16 data types will now work in /PROCESS=ODBC and /CSV. A complete list of source-level changes is available on request.

d) TYPE Parameter

The TYPE= specification in /FIELD statements is now supported and encouraged in new SortCL scripts to precede the data type declaration. Existing job scripts missing an explicit TYPE= precedent will work, and /FIELD statements with no TYPE still default to the ASCII data type.

e) EXT_FIELD Expansion

New format-specific, original column definitions at the end of /FIELD statements are now supported, encouraged, and automatically built in IRI Workbench-generated DDF specifications. Where EXT_FIELD definitions were only used for /PROCESS=ODBC or XDEF for XML before, now ODEF, XDEF, LDEF (for LDIF), JDEF (for JSON), and MDEF (for MongoDB) define original key name and value locations in syntax specific to each source format. This allows the SortCL field names to remain simple, case-insensitive, format-agnostic and easily mapped while improving metadata precision and lineage. Older EXT_FIELD syntax is supported for existing legacy scripts, which should be updated to the newer syntax.

f) UTF8 Handling Improvements & Multi-Byte Data Type Renaming

Conversion between the CoSort data types UTF8 and UTF16 has been added, and UTF8 to ASCII conversion for characters existing in both sets. SortCL scripts can be UTF-8 encoded, including field and condition names, literal values, separator and frame values. Previous "form 0" types with a conforming "form 6" type have become "form 6" in CoSort 10. UTF16 is now equivalent to UTF16_UNICODE, and the UTF32 data type has been removed. BIG5 is now equivalent to CHINESE_BIG5. GBK is now equivalent to CHINESE_GBK_SIMPLIFIED. SJIS is now equivalent to JAPANESE_ALPHABET. HANGUL and HHANGUL are now equivalent to KOREAN_HANGUL.

g) ODBC Query, Update and Delete

In addition to APPEND and INSERT introduced in 9.5, SortCL also supports SQL SELECT statements as another way to filter input, and /UPDATE and /DELETE to modify or remove data going to target tables.

h) /STREAM

New SortCL action statement for specification between /INFILE and /OUTFILE sections for processing streaming data from URLs.

i) HTTP and HTTPS Data Sources

Quoted URL strings can be specified as /INFILE values to indicate the location of an input file or data stream on the web. For example, /INFILE="http://host.company.com/path/employee.txt" and the /PROCESS must match the input data format.

j) FTP and SFTP Data Sources and Targets

Support for reading and writing files directly from FTP and SFTP, e.g., /INFILE="sftp://account-user:userpwd@host.company.com/path/employee.txt" /OUTFILE="ftp://anonymous@host.company.com/path/employee.txt" See the documentation on SFTP key assumptions and dependencies.

k) /PROCESS=JSON

Transforming, converting, masking, and reporting from and to JSON files is now supported. SortCL can thus read, process, and write flat and non-flat JSON objects, processing all non-flat objects through their flat values. Output layouts can be flat or non-flat.

l) "ismissing" Function

This new SortCL /FIELD function for JSON and future-supported semi and unstructured data sources tests whether a specified field's value path (in the JDEF) exists in the current record, even if the value is empty. ismissing can be used for /INCLUDE /OMIT in /INFILE or /INREC. ismissing can be used in /FIELD "IF" statements only in INREC. In /OUTFILE or with non JSON input it will always return false.

m) /PROCESS=MongoDB

SortCL now supports native MongoDB connections to MongoDB collections for reading, processing and writing non-binary document data.

n) S3 & HDFS Support

Read, process, and write files in Amazon AWS Simple Storage Service and Hadoop File System expressed as URLs in SortCL /INFILE and /OUTFILE.

o) MQTT & Kafka Support

Subscribe to, process, and publish messages through these brokers, defined in SortCL /INFILE and /OUTFILE URLs.

p) BOM support

Byte order marks are automatically detected in big and little endian files, eliminating the requirement for a /BOM statement under /INFILE.

q) SET file improvements

Set lookups can now use an empty value; i.e., a set item and can return an empty value if the set file column is empty and can search for an empty column. Set files can now also be used with UTF-8 type data.

r) Eclipse "Neon" Upgrade

IRI Workbench supporting CoSort 10 and other IRI tools is now based on Eclipse Neon and 64-bit JRE 1.8.

s) Random Noise

Data blurring for ages and ISO date is available as another data masking function to generalize PHI for HIPAA Expert Determination Method compliance.

t) Date Format Masking & Conversion

SortCL jobs can perform advanced date/time format conversion and value calculation using the change_dt /FIELD function and format specifiers.

u) GZip File Compression Level

To achieve a balance of speed and storage when writing output in gzip format, the ZIP_OUT_LEVEL tuning parameter sets the level (0-9, default 6) of target file compression. 0 is no compression. 1 is the fastest compression and 9 does the most compression, but at the slowest speed.

v) Null String Specification

A /NULL_STR= character replaces NULLs in database input with a specific string for SortCL processing, and to indicate to a target table when a value containing that string should go into the table with a NULL value.

w) Proxy Coupling Support

SortCL jobs can run seamlessly from DF-SORT and compatible JCL sort commands run from z/OS through Proxy Coupling Technology available from Proximal Systems Corporation.

x) Global Audit and Continuous Log Files Disabled

Running the cs_setup program no longer automatically creates the self appending XML file that can consume disk and tax SortCL runtime after many executions. The cosort.log file is also not enabled by default. Without the LOG entry, the .cserrlog file will be written in the current working directory for each execution if permitted. The LOG entry can be added if a record of each sortcl execution is desired.

y) /MEMORY-WORK Deprecated

This SortCL parameter is no longer supported nor encouraged, and will be ignored in existing SortCL scripts. See tuning options in Appendix D of the User Manual & Programmer's Guide.

z) SortI Deprecated

The Sort Interactive (sorti) utility is no longer shipped, documented, or supported in CoSort 10. For previous CoSort version users of sorti upgrading to this release, IRI can furnish a free sorti2scl utility to convert all valid .spc files to equivalent, extensible .scl scripts.

**(5) KNOWN ISSUES AND WORKAROUNDS**

The following limitations apply to the build accompanying this file. Therefore, the information below is subject to change without notice. Please contact your IRI representative or support@iri.com for updates.

a) DDF for JSON files and MongoDB collections

A standalone command-line utility for automatically generating SortCL data definition files does not accompany the initial release of CoSort 10. A DDF is instead generated in the IRI Workbench metadata discovery dialog. Note that JSON output is formatted using the JSON path specified in the output section of the script. When creating or modifying the JSON output format, be sure to specify JSON paths in their valid sequence. That is, there can be no duplicate value paths or path to a non-flat value. If fields in the output section are not in a valid sequence, invalid JSON output format will be produced.

b) 64KB record length limitation

All SortCL process types currently support records up to 65,535 bytes. In cases where the number of input files and the length of the input field exceed this limit, the file can still be processed with a specially compiled version (available upon request), and sort keys must be in the first 64KB. In the case of JSON, those larger files can also be processed if the SortCL script selects a subset of value paths that does not exceed the maximum input record length.

c) Order of expression

When cross-calculations involving functions are specified in SortCL /FIELD statements in an order other than as documented, a syntax error will be issued rather than allowing the expression to produce an incorrect result. Further development may support alternate orders.

d) Mixed Record Formats in Multi-Table Joins

When you have have different record formats (i.e., fixed vs. floating) in joins involving three or more files, sortcl completes the join only when the last-named join file is the only input source with a different record format. Workaround: Pre-process your differently formatted input sources so they are all in the same format prior to performing a join on three or more inputs.

e) Unsupported Formatting Options in Join Scripts

/INREC is not supported. Workaround: Specify input files needed only for join conditions and output display purposes. /HEADREAD and /HEADWRITE and /TAILREAD and /TAILWRITE cannot be used in a join because they are currently being processed. Workaround: Remove header and footer records in prior processing steps and reapply them if needed in a subsequent job script.

f) Overdefined Fields in /INREC

The derived_name=field_name convention is not supported in the INREC phase when the derived name is referenced in the output. Note that this does not affect the use of a derived_name=field_function or expression. Workaround: Since the feature is typically used to over-define the same field, redefine the field with the derived_name in the /INFILE section. A derived field name should not be given the same name as a source field name so as to prevent a circular reference error.

g) Unsupported BLOB and CLOB Columns

A later CoSort 10 update will contain handling provisions for these columns for database and file based operations.

h) Record Loss in ODBC /CREATE Operations

Rows read from and written to the same table will be lost when /CREATE and /REPORT are specified in a SortCL job. Changing either the job action to /SORT (which can change the order of the rows), or the write statement to /UPDATE will resolve this. Note also that /APPEND is the default behavior for /PROCESS=ODBC operations, while /CREATE is the default for files (where the above issue/workaround does not apply).

**(6) COPYRIGHT AND TRADEMARK ATTRIBUTION**