

CoSort Version 9.5

Big Data Manipulation

Software Overview, Interface Samples, Technical Specifications



TABLE OF CONTENTS

| | |
|---|----|
| AN INTRODUCTION TO COSORT..... | 3 |
| COSORT APPLICATIONS..... | 4 |
| COMPATIBLE APPLICATIONS..... | 6 |
| COSORT CONTENTS..... | 8 |
| SORT CONTROL LANGUAGE PROGRAM (SORTCL)..... | 10 |
| IRI WORKBENCH..... | 26 |
| SORTCL COMMAND SET..... | 30 |
| METADATA CONVERSION TOOLS | 31 |
| UNIX SORT REPLACEMENT (BIN/SORT)..... | 32 |
| SORT INTERACTIVE PROGRAM (SORTI)..... | 34 |
| COBOL MIGRATION TOOLS..... | 36 |
| APPLICATION PROGRAMMING INTERFACES (APIs) | 37 |
| SYSTEM TUNING..... | 39 |
| TECHNICAL SPECIFICATIONS..... | 41 |
| LICENSING INFORMATION..... | 48 |
| PROFESSIONAL SERVICES..... | 49 |
| COMPANY BACKGROUND..... | 49 |

AN INTRODUCTION TO COSORT

Since 1978, CoSort has been meeting the growing data manipulation needs of companies with high-volume flat-file, database and data warehouse installations. CoSort is also a favored solution for legacy sort and data migrations to Unix and Windows. IRI has worked to make CoSort the most widely licensed commercial sort product on open systems, and is heavily focused on the development of related data manipulation technologies.

CoSort is now a performance-enhancing solution for many applications, and a single-pass platform for large-scale:

| | |
|--------------------------|--|
| Data Processing | Transformation, Migration, Cleansing, etc. |
| Data Presentation | Reporting and Preparation |
| Data Protection | Data Masking, Encryption, etc. |
| Data Prototyping | Test Data Generation |

Data Processing

CoSort's Sort Control Language (SortCL) program can execute parallel data transformations to integrate, stage, and convert large data volumes. In just one I/O pass and job script, SortCL can:

select, sort/merge, join, lookup, convert data types and endian, file formats, re-map/reformat, pivot, sequence, calculate, aggregate, manage sub-strings, scrub, encrypt, de-identify, and perform complex transforms

Sources and targets include compressed, flat and index files, pipes, tables via Open Database Connectivity (ODBC), and custom procedures.

Data Presentation

SortCL users can output the results of the above processes into one or more detail and summary reports. Users can combine joins, cross-calculations, hash lookups, and conditional selection to generate formatted reports and subsets for: billing operations, customer segmentation, change data capture, forensic data analysis, and business intelligence tools. Formatting may include special field and file layouts, headers and footers, page numbers, environment values, embedded HTML tags (for web posting), and the conversion of data into CSV or XML for handoffs to BI and analytic tools.

Data Protection

SortCL, and the spin-off data masking product FieldShield, can secure sensitive data at the field level, based on business rules. Functions include 256-bit AES format-preserving encryption and de-identification, and data masking techniques to anonymize, obfuscate, pseudonymize, or redact fields. Additional encryption or security functions are also available through custom field transforms. Securing data at the field level (during or after processing and presentation) is faster, and leaves non-sensitive file, disk and database data available.

Data Prototyping

SortCL, and the spin-off test data product RowGen, can randomly create or select test field data and display it in real (production) file and report formats. You can create any number, type, and size of files, records and value ranges necessary to safely simulate reality and stress-test applications. Uses include database and ETL tool population, benchmarking, application development, and outsourcing.

COSORT APPLICATIONS

CoSort is a general-purpose, high-performance processor of sequential data in a variety of formats. It also serves as a migration platform for legacy data and sort programs, while also supporting business intelligence, ETL, and data governance operations. CoSort reduces runtimes, risks, and complexity for a variety of IT and business users.

| Job Function | CoSort Deliverables and Benefits |
|---|--|
| <i>IT Manager</i> | <ul style="list-style-type: none"> • Legacy sort software migration and modernization tools • Universal flat-file format, and data type, conversion capabilities • Detail and summary batch reporting with optional dashboard • Codes and runs faster than Perl, shell, SQL, ETL and COBOL jobs • Affordable price points and flexible licensing policies |
| <i>DBA</i> | <ul style="list-style-type: none"> • Parallel pre-sorts improve load, reorg and query performance • Combined sort, join, and aggregate transformations off-line • External batch and delta reports that are faster/easier than SQL • Flat-file lookups offer discrete, off-line, one-to-many solutions • Shared metadata with Fast Extract, FieldShield and RowGen |
| <i>Data Warehouse (ETL) Designer</i> | <ul style="list-style-type: none"> • Plug-in sort accelerators for DataStage and Informatica • Multi-threaded transformations in the file system for data staging • Complex selection and expression logic for data integration • Easy, open metadata and converters interface with existing tools • Integrated protection, custom transforms, and in/out procedures |
| <i>BI Architect</i> | <ul style="list-style-type: none"> • Big aggregation and filtering franchises data for BI tools • Embedded reporting with many formatting functions, including PCRE • Field-level protections enable compliant segmentation reports • Web log and IPA data handling facilitate click-stream analysis • Change data capture (delta) reporting using joins and selection |
| <i>CISO, Data Governance, or Compliance Officer</i> | <ul style="list-style-type: none"> • Field-level anonymization, de-ID, encryption, pseudonymization • Protection functions can run within transform and reporting jobs • Query-ready XML audit log of job details help verify compliance • Quality and safety improvements for Master Data held in flat files • Support for 24 of 34 COBIT 4.0 control objectives |
| <i>Application Developer (ISV)</i> | <ul style="list-style-type: none"> • Thread-safe API libraries for embedded parallel sorts, transforms • Serial and parallel system calls to the SortCL program • Access to included encryption libraries protect real-time data flows • Built-in test data generation capabilities ("RowGen" functionality) • Affordable licensing, customized to individual business models |

Figure 1 on the following page depicts the CoSort product flow diagram for fast, single-pass data manipulation.

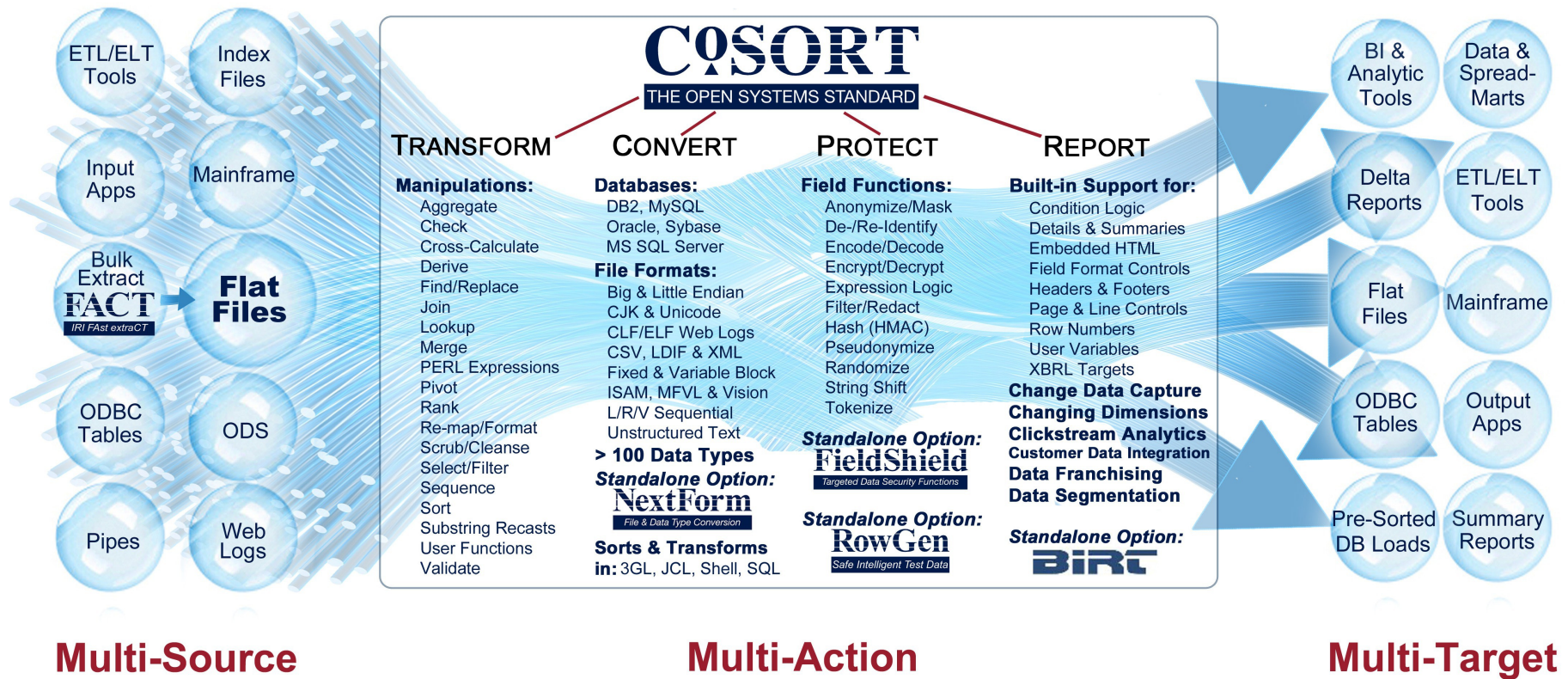


Figure 1 Fast Single-Pass Data Manipulation

COMPATIBLE APPLICATIONS

These other IRI tools create or leverage the metadata of CoSort's SortCL program:

| | |
|--|--|
| Fast Extract (FACT) | Unloads large database tables to flat files in parallel for off-line archival, data transformation, reporting, migration, and reloads. FACT can work through metadata and pipes with CoSort's SortCL tool to perform fast reorg, replication, encryption, ETL, and BI operations, all in one I/O pass. |
| Data masking (FieldShield) | De-identifies personally identifying information and other sensitive data residing in ODBC-connected database tables or popular file formats with multiple techniques such as encryption, obfuscation, hashing, pseudonymization, randomization, and tokenization. |
| Test Data (RowGen) | Creates safe test data in real file, report, and table formats for DB population, application development, benchmarking, etc. The same SortCL scripts used to process real data can be used to generate test data in the same formats. |
| Data Migration (NextForm) | Converts file formats, field data, and endian types for data, database, application, and platform migration projects. |

The following are examples of third-party products with which IRI maintains various levels of compatibility to enhance their operational performance or personal data privacy:

| | |
|--|--|
| Oracle | CoSort can source and target Oracle tables via ODBC or via OCI to the file system (using FACT and SQL*Loader). On this data, CoSort can: <ul style="list-style-type: none"> • transform (sort, join, aggregate, reformat, etc.) • capture changed data (generate delta reports) • protect (column-level encryption, masking, etc.) • perform index pre-sorts (on the longest table key) By pre-sorting, CoSort can improve the speed and efficiency of SQL*Loader operations, and thus reorgs and queries on the pre-CoSorted tables. With the space and time saved by offloading transforms and speeding loads, DBAs can also create and maintain more tables in optimal query order. |
| IBM DB2 UDB | CoSort can source and target DB2 tables via ODBC or the file system (using FACT and DB2 Load), to do the jobs above. The CoSort Load Accelerator for DB2 (CLA4DB2) speeds bulk loads. |
| IBM InfoSphere DataStage | CoSort's unique sort stage plug-in for DataStage Server Edition can improve sort performance up to 10 times with no interface changes. Subsequent join, aggregation and load stages benefit. Alternatively, running CoSort's SortCL program in the sequential file stage can enhance DataStage sort, join, and aggregation goals by running large transforms in fewer, faster, external passes. In SortCL, these transforms can also be combined with conversion, reporting, field protection, and load functions at the same time. |
| Informatica PowerCenter | CoSort's unique custom transform (CT) for PowerCenter's sort seamlessly replaces the native SorterTx component to improve its performance. Subsequent join, aggregation and loads benefit. IRI recommends, however, performing "push out" optimization of large transformation jobs by allowing CoSort's SortCL program to perform them in the file system. This approach is more efficient than "push-down optimization" into Oracle (DB layer transforms) and far less expensive than Teradata, AbInitio, DMEexpress, etc. |

BIRT

Business Intelligence Reporting Tool (BIRT) is the open source reporting system built on Eclipse. CoSort prepares (franchises) big, structured data sources for BI rapidly, with job design and management handled easily in Eclipse. And BIRT supports CoSort-prepared data directly, at reporting time.

Specifically, SortCL prepares data in disparate file and database sources for BI, by filtering, cleansing, sorting, joining, aggregating, masking, and formatting the data. For BIRT users, SortCL can:

- produce flat (CSV) files that BIRT can rapidly ingest; and/or
- directly transfer its data results in real-time to industry-driven BIRT templates through ODA in Eclipse

This means data integration normally performed on the IT side can now join BI report designers on the business side. This more seamless participation by both sides enables:

- better data profiling possibilities - to find the sources and relationships between data that matter to BI
- fast data integration and transformation - to feed precisely, properly-formatted data subsets to BI tools
- data quality and validation - because "good data in" means "good data out"
- data federation (virtualization) - for ad hoc BI use
- easy, on-the-fly integration and reporting - for what-if analysis
- data masking during transformation - to safeguard PII in reports

COSORT CONTENTS

The **CoSort** package contains standalone utilities, file layout metadata and sort parameter converters, third-party sort replacements, API libraries, and documentation.

The core **utility programs** in CoSort are:

- **SortCL** – the fourth-generation *sort control language* program for defining data and manipulations with syntax and semantics familiar to both mainframe sort and SQL users. The most comprehensive interface in the CoSort package, SortCL combines multiple data transformation functions (such as sorting, joining, aggregation and re-mapping) with reporting and field-level protections for: file compares and changed data capture, data type and file format conversions, data warehouse integration and staging (ETL), business intelligence, delta and summary reporting, data and database migration, data replication, federation, and compliance with data privacy regulations.
- **Sort** – a drop-in replacement for the Unix *sort* command that runs faster and scales linearly. It runs on all Unix and Windows (unixsort.exe) platforms.
- **SortI** – a user-friendly *sort interactive* session for simple sort/merge jobs. SortI offers context-sensitive help during the specification of ad hoc and batch jobs.

SortI and SortCL recognize environment variables and support pipes to allow data to flow between processes without additional I/O. SortI and SortCL may also be customized with user exits; such as procedures for special input, output, or comparison criteria. Usage instructions for each utility are in the user manual and man pages.

These **metadata converters** leverage your existing input and output layouts:

- **cob2ddf** translates COBOL copybook layouts to SortCL data definitions
- **csv2ddf** translates Microsoft **.csv** file headers to SortCL data definitions
- **ctl2ddf** translates Oracle SQL*Loader control file layouts to SortCL data definitions
- **elf2ddf** translates web logs in W3C "Extended Log Format" to SortCL data definitions
- **ldif2ddf** translates LDIF layouts to SortCL data definitions
- **xml2ddf** translates XML formats to SortCL data definitions
- **odbc2ddf** converts database table layouts into a SortCL data definition file (.ddf)
- **MIMB** (from MITI) translates many application file layouts to SortCL data definitions

These **sort parameter conversion utilities facilitate legacy sort migrations:**

- **mvs2scl** translates MVS JCL sort cards to SortCL job specifications
- **sorti2scl** translates SortI parameters to SortCL job specifications
- **vse2scl** translates VSE JCL sort cards to SortCL job specifications

IRI Workbench, a graphical user interface (GUI) built on Eclipse, facilitates the specification, execution, tuning and maintenance of SortCL job scripts through job creation and metadata definition wizards, a dynamic job outline, and a syntax-aware editor for manual SortCL specification.

The IRI Workbench also provides database data access, viewing, and integration into SortCL and all other IRI software product jobs. It includes extensions for team contributions, job version control, and remote system data sourcing and job execution.

The following third-party **sort replacements** are available with the CoSort package:

- **acu-cosort** – a drop-in replacement for the sort verbs supplied with ACUCOBOL-GT
- **cla4db2** – the “CoSort Load Accelerator for DB2” replaces IBM’s sort routine within the UDB 5-8 loader, as much as doubling throughput on Unix
- **mf-cosort** – a drop-in replacement for the sort verbs supplied with Micro Focus Net Express, Server Express, and Workbench on Unix and Windows. COBOL users can link statically or dynamically to mfcosort and the CoSort engine to accelerate sort speed and reduce temporary sort space in new executables or a full RTS
- **nat-sort** – a drop-in replacement for the sort verb in Software AG Natural
- **proc-sort** – SAS System 7-9 users can link dynamically to shared cosort() libraries to replace the sort function in SAS on Unix systems
- **Sorter Tx CT** – a drop-in custom transform for the sort in Informatica PowerCenter
- **Sort PlugIn** – a drop-in replacement for the IBM Infosphere DataStage SE sort.

CoSort provides similar drop-in sort replacement facilities for Tetrad OPX, and the UniKix Mainframe Batch Manager. CoSort hooks are also available for The ETI Solution, Kalido’s Dynamic Information Warehouse, and IDS Korea’s TeraStream ETL suite.

For developer and Independent Software Vendor (ISV) use, CoSort also includes two Application Programming Interface (API) libraries.

- **cosort_r()** – A thread-safe version of the original cosort() API that allows multiple coroutine sort/merge operations to occur in the same pass through the data. The coroutine engine allows in-memory record transfers between programs and the sort.
- **sortcl_routine()** – The thread-safe SortCL library that allows programmers to exploit the full range of CoSort Sort Control Language commands within their own programs.

CoSort APIs let you define any input (selection), compare (order sequence), or output (reporting) criteria, enabling applications to accomplish complex jobs in one I/O pass. You can write calls to either library in any language that can link to a C library, such as C++, COBOL, VB, Java, etc.

Finally, the CoSort package also includes the following **documentation**:

- **Installation Guide** – platform-specific loading, licensing, and configuration advice
- **Manual** – full user and programmer documentation for all the above interfaces
- **Job Examples** – sample SortCL job scripts, metadata conversions, and API calls
- **Best Practices** – tips and tricks for job scripting and tuning

SORT CONTROL LANGUAGE PROGRAM (SORTCL)

Well beyond traditional sort/merge operations, **CoSort** provides a broad range of big data manipulation and management functions for data warehousing, legacy migration, and business intelligence projects through its fourth-generation Sort Control Language (SortCL) program. SortCL is a language, program, tool and library that allows end-users and developers, working with simple text scripts or an Eclipse GUI, to:

| Function | Actions |
|---------------------------------|--|
| Filter | At the byte, field and record level, plus duplicate removal and saving |
| Segment | Conditional (include/omit) selection with if-then-else, else-if logic |
| Sort | Multiple keys, directions, sequences |
| Merge | Two or more pre-sorted files |
| Join (Match) | Two or more un/sorted files on many conditions for file compares and change data capture (deltas) |
| Aggregate | Parallel roll-up and drill-down sum, min, max, average, and count values. Accumulate (Running). Rank. Lead and Lag (Windowing). |
| Check | Verify source data is pre-sorted prior to sort or join operations |
| Re-Map | Resize, reposition, and realign fields |
| Convert | Change data types (e.g. EBCDIC<>ASCII, Packed<>Numeric) |
| Re-format | Convert between file formats (e.g. Text <>XML, VS<>RS, ISAM<>Vision, LDIF<>CSV) |
| Pivot / Unpivot | De-normalize and normalize dimensional layouts |
| Cleanse | De-duplicate, validate, homogenize, filter, find/replace, and re-structure |
| Enrich | Integrate and segment data enhance row and column detail. Create new data forms and layouts through conversions, calculations and expressions and composite (templates). |
| Migrate DBs | via remapping |
| Calculate | Math and trig functions across detail and summary rows |
| Sub-string | Bit-level manipulations and Perl-compatible regular expression logic for pattern matching, etc. |
| Validate | Check that character and field attributes match their specifications (i.e. "iscompares", gap analysis) |
| Sequence | For custom indexing, reporting, and database load operations |
| Set Lookup | Discrete field substitutions, pseudonymization, etc. using "set" file field dimensions |
| Fuzzy Lookup | For slowly changing dimension (SCD) reporting |
| Federate | Get discrete (lookup) values and virtualize results in via reports and replicas |
| Protect | Encrypt and mask data at the field level and audit data security measures. Also: anonymization, de-identification, filtering, and pseudonymization |
| Test | Create randomly-generated or set-selected (safe) test data fields |
| Report | Custom-formatted, segmented detail and summary targets |
| Replicate | Copy, manipulate and move data from one or more sources to one or more targets |
| Custom | Complex field-level user functions (e.g., 3rd-party DQ libraries) |

SortCL can support the combination of these functions all in one job script and I/O pass through multiple data sources and targets.¹

By running multiple data manipulations at once, you can use SortCL to:

- Package, protect, and provision big data without Hadoop
- Filter, integrate, and stage data for DW, ODS, BI, data marts, and spreadmarts
- Replace slower 3GL, shell, Perl and SQL procedures
- Transform high volumes outside BI, DB and ETL tools
- Relieve application and system overhead
- Generate custom detail, delta, and summary reports
- Accelerate bulk database reorgs and loads
- Detect, capture, and audit changed data
- Consolidate data privacy with transformation and reporting

SortCL uses a self-documenting Data Definition Language (DDL) and Data Manipulation Language (DML) syntax that is familiar to both mainframe sort and SQL users. SortCL DDL repositories or data definition files (DDF) can be centralized and re-used. By supporting the separation of data definition and manipulation statements, SortCL supports the use of shared metadata, and the independence of data from applications.

SortCL job scripts can be invoked from: the command line, the IRI Workbench, a batch processing stream, or the thread-safe SortCL API (sortcl_routine).

Application-level statistics can be output with each job, either to the screen or a file. In addition, the CoSort job log runs in a self-appending file, and sends debugging information into a self-replacing file. On-screen monitoring options are available at various verbosity levels for runtime progress assessment. You can also enable and secure an XML audit log file to validate compliance and perform forensic application and data analysis.

Users converting from legacy sort products can leverage CoSort's metadata conversion tools and services to ease job script migrations to SortCL. See METADATA CONVERSION TOOLS.

SortCL Operations

One of the most basic SortCL scripts that you can write contains only an infile and an outfile, as in the following:

```
/infile=accts695  
/outfile=accts695.new
```

This simply sorts the accounts695 file from left to right without reformatting.

SortCL processes data in three phases -- input, action (processing), and output. In the input phase, source records are processed with selection. Actions are sort, merge, join, report, or check. In the output phase, selected records are remapped to one or more targets simultaneously. Derived fields, aggregates, additional filters, and multiple formats can be conditionally defined in the same target. A special "inrec" section is defined when a virtual record layout is needed for processing input sources that are formatted differently.

¹ With IRI's Fast Extract (FACT) unload tool, bulk input can also come directly from RDBMS tables.

SortCL also has the ability perform and combine many more data transformations, as well as protect data at risk, and produce formatted reports, all at the same time. Through the use of metadata repositories -- SortCL DDFs -- you can define and share any structured data subset, or relational view, in the SortCL job specification files (.SCL) that reference the DDFs. Those file layouts can be re-used in many applications.

SortCL Application Sample #1 – Sort and Reformat, Metadata Repository

This SortCL job is a simple two-key sort job. A single, flat input file is specified directly in the script. In addition to re-ordering the data, this script will convert the file layout from a pipe-delimited format to fixed position fields. Notice, however, that the fixed output field definitions are stored in this reusable SortCL “data definition file” metadata repository:

Data Definition file “chiefs.ddf”

```
/FIELD=(president,POS=1,SIZE=22)
/FIELD=(service,POS=25,SIZE=9)
/FIELD=(state,POS=40,SIZE=2)
/FIELD=(party,POS=45,SIZE=3)
```

Input file 1, “chiefs_10_sep”

```
Eisenhower, Dwight D.|134|1953-1961|REP|TX
Kennedy, John F.|135|1961-1963|DEM|MA
Johnson, Lyndon B.|136|1963-1969|DEM|TX
Nixon, Richard M.|137|1969-1973|REP|CA
Ford, Gerald R.|138|1973-1977|REP|NE
Carter, James E.|139|1977-1981|DEM|GA
Reagan, Ronald W.|140|1981-1989|REP|IL
Bush, George H.W.|141|1989-1993|REP|TX
Clinton, William J.|142|1993-2001|DEM|AR
Bush, George W.|123|2001-2009|REP|TX
```

```
### CoSort SortCL Job Specification ###

### Input Phase ###
/INFILE=chiefs_10_sep
  /FIELD=(president,POSITION=1,SEPARATOR='|')
  /FIELD=(votes,POSITION=2,SEPARATOR='|')
  /FIELD=(service,POSITION=3,SEPARATOR='|')
  /FIELD=(party,POSITION=4,SEPARATOR='|')
  /FIELD=(state,POSITION=5,SEPARATOR='|')

### Action Phase ###
/SORT
  /KEY=party
  /KEY=president

### Output Phase ###
/OUTFILE=chiefs.out
  /SPECIFICATION=chiefs.ddf # metadata
```

As shown above, the job script on the right uses explicit layouts for the input, but relies on the metadata file, **chiefs.ddf**, for the output layout. By centralizing the metadata, it can be used in other SortCL job scripts.

SortCL job scripts are typically run from a batch script, with a command entry similar to:

```
$COSORT_HOME/bin/sortcl /SPECIFICATION=/path2/example1.scl
```

Output file 1, "chiefs.out"

| | | | |
|-----------------------|-----------|----|-----|
| Carter, James E. | 1977-1981 | GA | DEM |
| Clinton, William J. | 1993-2001 | AR | DEM |
| Johnson, Lyndon B. | 1963-1969 | TX | DEM |
| Kennedy, John F. | 1961-1963 | MA | DEM |
| Bush, George H.W. | 1989-1993 | TX | REP |
| Bush, George W. | 2001-2009 | TX | REP |
| Eisenhower, Dwight D. | 1953-1961 | TX | REP |
| Ford, Gerald R. | 1973-1977 | NE | REP |
| Nixon, Richard M. | 1969-1973 | CA | REP |
| Reagan, Ronald W. | 1981-1989 | IL | REP |

The input file, **chiefs_10_sep** is now in order by party and president, and displayed according to the fixed position layout specified in **chiefs.ddf**. Notice that the second input field, *votes*, was not in the output file specification, and that the state and party field were transposed. By mapping using symbolic field name references, SortCL gives you field level control of all of your output targets, as the following examples will further demonstrate.

SortCL Application Sample #2 – Data Transformation and Protection

This SortCL job is an example of a single-key sort. The fields are defined in the input phase, the sort key in the action phase, and then, in the output phase a series of target files are defined in different formats for different departmental purposes. Notice how individual fields are protected according to different business rules, or role based access controls (RBAC).

The input file below was generated with IRI's RowGen tool to create realistic transaction data. Note that any number of sources can be input and that these input sources can have any number of formats. The input sources can be files, pipes, and/or procedures. Data integration of this kind was not demonstrated for the sake of simplicity.

Input file, "seqdata"

| | | | | | | | | | |
|----|-----------|------------------|------------|---|----|-------|---------|-----------|---|
| 01 | 330170363 | Stuart Clay | 0056681.42 | 6 | cT | 101 | B | St | B |
| 02 | 421901269 | Taylor Guerrero | 0015019.10 | 9 | MD | 1031 | Park Ln | Apt D | A |
| 03 | 529433545 | Charles Caldwell | 0041116.71 | 3 | NY | 14 | Main | St | A |
| 04 | 129737773 | Robyn Puckett | 0044558.62 | 3 | ny | 822 | Hwy | 76 | B |
| 05 | 594521240 | Santiago Lindsey | 0055836.11 | 0 | TX | Star | Rt | Box 822 | A |
| 06 | 796569799 | Charles Lindsey | 0098525.58 | 2 | TX | 12746 | Wolf | Circle | A |
| 07 | 384127387 | Santiago Puckett | 0059036.80 | 4 | NY | 321 | Baltic | Ct | B |
| 08 | 711604065 | Charles Williams | 0018645.95 | 1 | Tx | 1103 | Fresh | Creek Ln | A |
| 09 | 343054521 | Jack Velazquez | 0029205.44 | 2 | NY | 6780 | Sand | Dr Apt 3A | B |
| 10 | 148354977 | Donald Cooke | 0044121.44 | 4 | MA | 35 | La | Palma Dr | A |

The following job script produces several output files in the same job script and I/O pass, which includes all 10 records from the above input files (though filters could have been applied).

```

### Sort Control Language (SortCL) Program ###

### Input Phase ###
/INFILE=(seqdata)           # Reads 1 input file
/FIELD=(idnum,POS=1,SIZE=2)  # Unique record identifier tag
/FIELD=(ssno,POS=4,SIZE=9)
# Overdefines the social security number into parts for obfuscation
/FIELD=(ssno_part1,POSITION=4,SIZE=1)
/FIELD=(ssno_part2,POSITION=5,SIZE=4,NUMERIC)
/FIELD=(ssno_part3,POSITION=9,SIZE=4,NUMERIC)
/FIELD=(name,POSITION=14,SIZE=20)
# Defines the last name letter field for conditional selection
/FIELD=(last_name_letter,SEPARATOR=' ',POSITION=4,SIZE=1)
/FIELD=(salary,POSITION=36,SIZE=10,NUMERIC)
/FIELD=(salary2,POSITION=36,SIZE=10) # Redefined as ASCII for encryption
/FIELD=(deduction_no,POSITION=47,SIZE=1)
/FIELD=(state,POSITION=49,SIZE=2)
/FIELD=(address,POSITION=52,SIZE=20)
/FIELD=(group_code,POSITION=75,SIZE=1)
/FIELD=(wholerec,POSITION=1,SIZE=71)
# Define a field to be the entire record

### Action Phase ###
/SORT
/KEY=(group_code)           # Sort on code field for aggregation
/KEY=(salary)               # Sort on salary field

### Output Phase ###
/OUTFILE=testdata          # First Output File. Obfuscates data, preserves
format
/FIELD=(idnum,POSITION=1,SIZE=2.0,FILL='0',NUMERIC)
/FIELD=(ssno_part1,POSITION=4,SIZE=1)
# Obfuscate the next 4 digits through conditional cross-calculation
/FIELD=(ssno_part2_new,POSITION=5,SIZE=4.0,FILL='0',NUMERIC,\
    IF ssno_part2 GT 4500 THEN ssno_part2 / 2 ELSE 2 * ssno_part2 - 55)

# Obfuscate the final 4 digits
/FIELD=(ssno_part3_new,POS=9,SIZE=4.0,FILL='0',NUMERIC,\
    IF ssno_part3 GT 4500 THEN ssno_part3 / 2 ELSE 2 * ssno_part3 -
54)
# Create pseudonym of the real name using the lookup table pseudo.set
/FIELD=(name_fake,POSITION=14,SIZE=20,SET=pseudo.set[name])
# Make the salaries anonymous using conditional cross-calculation
/FIELD=(salary_new,POSITION=35,SIZE=10.2,NUMERIC,\
    IF salary GT 50000.00 THEN 0.85 * salary ELSE 1.15 * salary)
/FIELD=(deduction_no,POSITION=46,SIZE=1)
/FIELD=(state,POSITION=48,SIZE=2)
/DATA=" "
/DATA={20}"*"              # Masks address data with asterisks

```



```

/OUTFILE=aggregate_salaries_by_group      # Summary record format
/FIELD=(total_salary,POSITION=51,SIZE=12,CURRENCY)
/SUM total_salary FROM salary BREAK group_code
/OUTFILE=aggregate_salaries_by_group      # Detail record format
/FIELD=(group_code,POSITION=1,SIZE=1)
/FIELD=(name,POSITION=3,SIZE=20)
/FIELD=(address,POSITION=25,SIZE=20)
/FIELD=(TOUPPER(state),POSITION=47,SIZE=2) # Capitalization function
/FIELD=(salary,POSITION=51,SIZE=12,CURRENCY)

/OUTFILE=salaries_de_id
# De-identify fields using bit manipulation function (to preserve field size)
/FIELD=(idnum,POSITION=1,SIZE=2.0,FILL='0',NUMERIC)
/FIELD=(de_identify(salary,"abc"),POSITION=6,SIZE=10,NUMERIC)
/FIELD=(TOUPPER(state),POSITION=20,SIZE=2)

/OUTFILE=encrypt_all
/FIELD=(encryptAES256(wholerec),POSITION=1) # Default key phrase used

/OUTFILE=encrypt_2fields
# Encrypt 2 fields, each with a different key phrase
# To determine the size of the encrypted output field:
# Increase field size to next multiple of 16 and divide by
3
# Round up to the next whole number and multiply by 4
/FIELD=(encryptAES256(ssno,"passphr1"),POSITION=1,SIZE=24)
/FIELD=(name,POSITION=26,SIZE=20)
/FIELD=(encryptAES256(salary2,"passphr2"),POSITION=47,SIZE=24)
/FIELD=(TOUPPER(state),POSITION=73,SIZE=2)

/OUTFILE=report.csv
/PROCESS=CSV # Creates header from fieldnames
/FIELD=(idnum,POSITION=1,SEPARATOR=',',FRAME='')
/FIELD=(ssno,POSITION=2,SEPARATOR=',',FRAME='')
/FIELD=(name,POSITION=3,SEPARATOR=',',FRAME='')
/FIELD=(salary,POSITION=4,SEPARATOR=',',FRAME='')
/FIELD=(deduction_no,POSITION=5,SEPARATOR=',',FRAME='')
/FIELD=(state,POSITION=6,SEPARATOR=',',FRAME='')
/FIELD=(address,POSITION=7,SEPARATOR=',',FRAME='')
/FIELD=(group_code,POSITION=8,SEPARATOR=',',FRAME='')

```

This job script turns the two input files into six output files, all in one I/O pass. Many more inputs or outputs, of any size and format, could have been specified, and conditional selection criteria could have been applied against any source or target. A number of additional field level transformation functions could also have been specified. For a more complete list of available data manipulation functions that can be performed in SortCL, see the TECHNICAL SPECIFICATIONS chapter.

Outputs from the sample above follow, along with explanations of each.

Output file 1, "testdata"

```
02 443252484 Clifton Jimenez      17271.97 9 MD *****
08 722658076 Jeffery Gomez       21442.84 1 Tx *****
03 558317036 Teddy Black        47284.22 3 NY *****
10 124182488 Julio Koch         50739.66 4 MA *****
05 547262426 Spencer Craig      47460.69 0 TX *****
06 748284900 Landen Sullivan     83746.74 2 TX *****
09 385552260 Francisco Duffy     33586.26 2 NY *****
04 158913886 Salvador Jacobson   51242.41 3 ny *****
01 359790672 Ramsey Flynn       48179.21 6 cT *****
07 342063694 Alvaro Mcleod      50181.28 4 NY *****
```

This file shows safe, protected results in the form of similarly formatted test data. The data are sorted on group_code and then salary prior to the salary being protected; therefore, the salary order will not appear to be ordered. Note that the social security numbers were obfuscated with expression logic defined in 2 lines of the SortCL script Part 2 is shown below:

```
/FIELD=(ssno_part2_new,POSITION=5,SIZE=4.0,FILL='0',NUMERIC, \
      IF ssno_part2 GT 4500 THEN ssno_part2 / 2 \
      ELSE 2 * ssno_part2 - 55)
```

and that names were de-identified with pseudonyms in a look-up table called pseudo.set:

```
/FIELD=(name_fake,POSITION=14,SIZE=20,SET=pseudo.set[name])
```

SET file, "pseudo.set"

```
Charles Caldwell      Teddy Black
Charles Lindsey      Landen Sullivan
Charles Williams     Jeffery Gomez
Donald Cooke        Julio Koch
Jack Velazquez       Francisco Duffy
Robyn Puckett        Salvador Jacobson
Santiago Lindsey     Spencer Craig
Santiago Puckett     Alvaro Mcleod
Stuart Clay          Ramsey Flynn
Taylor Guerrero      Clifton Jimenez
/default/
```

In the above set file, the 'real name' Charles Caldwell is identified with the 'fake name' Teddy Black, and so on. A tab character separates the two names in the lookup table.

Output salaries were anonymized with math functions (which caused the appearance of disorder in that field). The TOUPPER syntax applied in the state field converted the input values to all upper case letters on output; this is one of several built-in data quality functions. Finally, the last three input fields were redacted and masked with asterisks.

At the same time, this output file (with protected real data) was also produced:

Output file 2, "aggregate_salaries_by_group"

| | | | |
|--------------------|---------------------|----|--------------|
| A Taylor Guerrero | 1031 Park Ln Apt D | MD | \$15,019.10 |
| A Charles Williams | 1103 Fresh Creek Ln | TX | \$18,645.95 |
| A Charles Caldwell | 14 Main St | NY | \$41,116.71 |
| A Donald Cooke | 35 La Palma Dr | MA | \$44,121.44 |
| A Santiago Lindsey | Star Rt Box 822 | TX | \$55,836.11 |
| A Charles Lindsey | 12746 Wolf Circle | TX | \$98,525.58 |
| | | | \$273,264.89 |
| B Jack Velazquez | 6780 Sand Dr Apt 3A | NY | \$29,205.44 |
| B Robyn Puckett | 822 Hwy 76 | NY | \$44,558.62 |
| B Stuart Clay | 101 B St | CT | \$56,681.42 |
| B Santiago Puckett | 321 Baltic Ct | NY | \$59,036.80 |
| | | | \$189,482.28 |

This is a detail and summary report, grouping records by their group code (A or B). Two same-named /OUTFILE sections were used: one to define the format of the detail records, and the other for the summary information. The group totals were derived with this syntax:

```
/SUM total_salary FROM salary BREAK group_code
```

Output file 3, "salaries_de_id"

| | | |
|----|------------|----|
| 02 | 335735<253 | MD |
| 08 | 335:9872<7 | TX |
| 03 | 33855592;5 | NY |
| 10 | 3388545288 | MA |
| 05 | 3377:69255 | TX |
| 06 | 33<:74727: | TX |
| 09 | 334<437288 | NY |
| 04 | 338877:294 | NY |
| 01 | 33799:5284 | CT |
| 07 | 337<3692:3 | NY |

This third output file contains the same information, but without the header record. Salaries have been de-identified with an internal bit manipulation function using the pass code 'abc' (which creates the bit manipulation parameters). This method is akin to, but less secure than, encryption, but preserves the field size.

Output file 4, "encrypt_all"

| |
|---|
| 1S7t5h+/s3sUbOs+42pYtkTau2K4kzZMOB4CrvE5QSJes2kvxrYIiWtg3y4VWYT7qIoc/rYwELAux4Gh3Cg9EoWf680dPh+ATqOJ4xE6/T4=Xh2nImJY4hBLfwWIDGeR1d0hNCC4Fbtz1UICR31wgr7rQo7byO1fVFGw5mwh+GSbh5OM7icopQ841NVnODVuw6QpqLJCipwkrYyJ00wvDFg=twoxGCKGxWn76wrnHlBMX7VlAdRAqXne1l23i301jkMhbWkf+K9E6JZ/iJtHElPi90bgeqGJyaXcMRMRHlcQkdV/wLAXJJ2b1N16Sz/A8uQ=O1v4mpIxb+D+egm3MbovHPilhn1MI67Kmdza38JvfIjEw5/tXLBFC4G1lUK1eaaNDYdRuJw8CcV1wf05nJlwQU5UmXmP8uIqeHW41tFK2Xo=b7B4w8SWnGam7gXFC6pGAulqOwp3aX8vFTkRSmUF57hFLQ9IxmLjd7nqDQKMgPg9SVBvlcB4Jsk13TIL6NKM5J0TCKYIH38aVfcw6cG+LuA=usnCRehsH109zkzDdCYJnfkTp35IjdgA+p4JXZAF3W4uPlghieH4fKg5zLxYZPtfa1ThT4oWxNCV9Yb/Ey4m/oTM8CaqFderHAQRQJiqj5c=e0DUXvL1/axHuLzgXPaCG0w/IT1QKcypks3a19hXCCHGH29yegi3Tp+9/nVGaWxVn/UFBYvuiVOJB1hw9RQuzmJMDsuEnr6zGbp14gvzrCA=ysO8dAWoGb2soHkp+WlpML7B2MXR8i5kKjzcHjCnxEgoU5Dtd/GXToy1fuM0CrfJfmz3F/oRnlw6YhVlvAc46X+Cr033y0y5eCIrATGFqGkI=Pe8gYs51/3z9eQrxobmMo/FHI0c1QboyNylmtWdWBa4/UekwCssPRnvYh8J17y4PrmE9V94WPTZ3RrL7rijQdz8WzS8ES1taZ/MpVaUJ0Gk=LR5H66hXx7YtaANSJRCiwmC2Hssc6YA3VahSRtUQDWTkF/+vxonCrpR6tP87LphMB/k3hKUD5rxuaJkCLrugfZhd3XiLGDxnuryfOXhPeJo= |
|---|

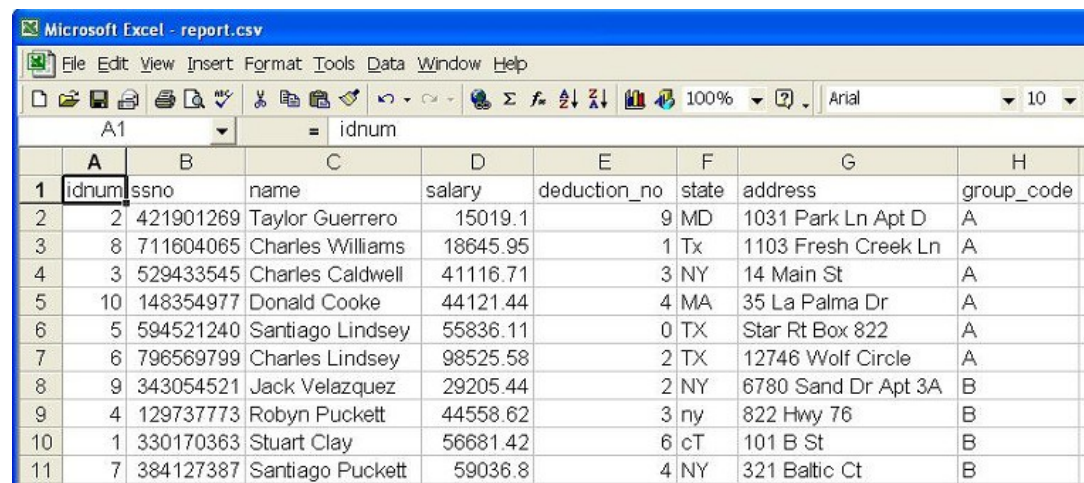
Here, the entire record was declared as a field and encrypted. You may have noticed that the encrypted “ciphertext” results contain only printable ASCII characters for printing and inclusion in text files.

Output file 5, “encrypt_two_fields”

| | | | |
|--------------------------|------------------|--------------------------|----|
| bUWn/CrhAql4MXNxoy70kA== | Taylor Guerrero | 2iyseYtWEd3ESFwE91433Q== | MD |
| 8XvhrNe7ecF2DKFZyP4cNw== | Charles Williams | SK8auRS56YxRWMtLTXEGtg== | TX |
| pHxUwHhkcXW06JEhoDilJg== | Charles Caldwell | bCz/W5bEpDpA2KJYDi3xvQ== | NY |
| sAol8DlsqS2viXEMPHullQ== | Donald Cooke | kpCZLODnoWFht50aH7u2LQ== | MA |
| eGoUczVoZEm95/9eBD8iOQ== | Santiago Lindsey | 8inKXdYDU4AH9tJx8xlPAg== | TX |
| AJJEKjJuJQErVR726ytkZg== | Charles Lindsey | Vzm4/kcz/ypUNfWJjBBrcA== | TX |
| 2KcftKVKDv+OCaG/FFdJ6w== | Jack Velazquez | m8MIpOmfpQxtNsO69cV0Mg== | NY |
| QMCKCxi32a9ZL3cYuBwHew== | Robyn Puckett | qwn2T7pUF/Hu+YPQBThcwg== | NY |
| CcxB9LuMkdAJ0E3rhwDiuA== | Stuart Clay | Q/bc3I756EPqn3TAYcqOUA== | CT |
| RQVeipLI4xlzvwzskHTE0Q== | Santiago Puckett | 9ilfjr3CQs0yjuKV/tVCEw== | NY |

The fifth output file, shown above, encrypts the social security number (SSN) and salary fields, but with two different pass phrases so the data is protected for different disclosures.

Output file 6, “report.csv”



| | A | B | C | D | E | F | G | H |
|----|-------|-----------|------------------|----------|--------------|-------|---------------------|------------|
| 1 | idnum | ssno | name | salary | deduction_no | state | address | group_code |
| 2 | 2 | 421901269 | Taylor Guerrero | 15019.1 | 9 | MD | 1031 Park Ln Apt D | A |
| 3 | 8 | 711604065 | Charles Williams | 18645.95 | 1 | Tx | 1103 Fresh Creek Ln | A |
| 4 | 3 | 529433545 | Charles Caldwell | 41116.71 | 3 | NY | 14 Main St | A |
| 5 | 10 | 148354977 | Donald Cooke | 44121.44 | 4 | MA | 35 La Palma Dr | A |
| 6 | 5 | 594521240 | Santiago Lindsey | 55836.11 | 0 | TX | Star Rt Box 822 | A |
| 7 | 6 | 796569799 | Charles Lindsey | 98525.58 | 2 | TX | 12746 Wolf Circle | A |
| 8 | 9 | 343054521 | Jack Velazquez | 29205.44 | 2 | NY | 6780 Sand Dr Apt 3A | B |
| 9 | 4 | 129737773 | Robyn Puckett | 44558.62 | 3 | ny | 822 Hwy 76 | B |
| 10 | 1 | 330170363 | Stuart Clay | 56681.42 | 6 | cT | 101 B St | B |
| 11 | 7 | 384127387 | Santiago Puckett | 59036.8 | 4 | NY | 321 Baltic Ct | B |

The last output file above, viewed in a spreadsheet, shows how specifying /PROCESS=CSV on output automatically adds a header record with the output field names. The conversion of the source records to a comma-separated framework (including the addition of the header record) allowed the output target to be read by Excel without additional processing.

SortCL Application Sample #3 – Alpha-Numeric Format-Preserving Encryption

This example uses personal information data, **personal_info**, including credit card numbers, driver's license numbers, and names.

| | | |
|---------------------|-------------------|------------------|
| 9654-4338-8732-8128 | W389-324-33-473-Q | Jessica Steffani |
| 2312-7218-4829-0111 | H583-832-87-178-P | Cody Blagg |
| 8940-8391-9147-8291 | E372-273-92-893-G | Jacob Blagg |
| 6438-8932-2284-6262 | L556-731-91-842-J | Just Rushlo |
| 8291-7381-8291-7489 | G803-389-53-934-J | Maria Sheldon |
| 7828-8391-7737-0822 | K991-892-02-578-O | Keenan Ross |
| 7834-5445-7823-7843 | F894-895-10-215-N | Francesca Leonie |
| 8383-9745-1230-4820 | M352-811-49-765-N | Nadia Elyse |
| 3129-3648-3589-0848 | S891-915-48-653-E | Gordon Cade |
| 0583-7290-7492-8375 | Z538-482-61-543-M | Hanna Fay |

The following SortCL script encrypts the credit card and driver's license number fields, while preserving the field formats.

```
### Sort Control Language (SortCL) Program ###

### Input Phase ###
/INFILE=personal_info      # Reads 1 input file
  /FIELD=(credit_card,POS=1,SEP='\t')
  /FIELD=(driv_lic,POS=2,SEP='\t')
  /FIELD=(name,POS=3,SEP='\t')

### Action Phase ###
/REPORT

### Output Phase ###
/OUTFILE=personal_info_encrypted
  /FIELD=(credit_card1=FPE_ALPHANUM(credit_card,"pass"),POS=1,SEP='\t')
  /FIELD=(driv_lic1=FPE_ALPHANUM(driv_lic,"pass"),POS=2,SEP='\t')
  /FIELD=(name,POS=3,SEP='\t')
```

This produces **personal_info_encrypted**:

| | | |
|---------------------|-------------------|------------------|
| 0832-9678-1911-0645 | R784-107-86-619-Q | Jessica Steffani |
| 0835-7171-0577-5699 | G156-454-45-303-O | Cody Blagg |
| 0789-2128-0461-5374 | Q305-118-71-384-Q | Jacob Blagg |
| 1591-0561-0417-5772 | D344-156-20-555-G | Just Rushlo |
| 9296-9613-4710-5436 | U751-860-67-075-Y | Maria Sheldon |
| 9881-4436-0773-0973 | X878-716-85-252-C | Keenan Ross |
| 4594-9802-2566-4840 | T273-579-67-063-M | Francesca Leonie |
| 6514-3079-6147-6828 | A617-849-83-864-X | Nadia Elyse |
| 9221-6125-6496-9606 | S039-406-12-369-U | Gordon Cade |
| 1404-8512-8389-2619 | K379-587-05-591-C | Hanna Fay |

SortCL Application Sample #4 – Data Transformation and Reporting

This example uses stock trading data. It performs a full outer join of two differently formatted input files to produce three differently formatted targets. The script includes cross-calculation, aggregation, selection, and markup tags for BI, data interchange, and web posting purposes.

It is important to remember that the samples shown are small in order to illustrate combinable functionality, not speed in volume. The major benefit of SortCL, in addition to task consolidation, is its fast processing of big data sources together.

In this example, the input files to be joined are unsorted. The first, **nyse-a**, is in tab-delimited format, an export from a DB. The second input file, **buys.csv**, is in CSV format, typical in spreadsheet applications.

Input file 1, "nyse-a"

| | | | | | |
|-------------------------|-----|--------|---------|------|------|
| A. O. Smith Corporation | AOS | 42.40 | 142900 | 0.04 | 0.09 |
| A.G. Edwards Inc. | AGE | 52.81 | 251800 | 0.48 | 0.91 |
| A.O. Tatn EFT First | TNT | 103.01 | 136000 | 1.01 | 0.99 |
| AAG Holding Company1 | GFZ | 24.71 | 1900 | 0.06 | 0.24 |
| AAG Holding Company2 | GFW | 25.05 | 4200 | 0.05 | 0.20 |
| Aames Investment Corp. | AIC | 4.84 | 145500 | 0.04 | 0.82 |
| Aaron Rents, Inc. | RNT | 24.05 | 1706300 | 2.53 | 9.51 |
| ABB LTD. | ABB | 12.55 | 2456100 | 0.13 | 1.04 |
| Abbey National plc | SXA | 25.00 | 24700 | 0.15 | 0.60 |
| Abbott Laboratories | ABT | 47.25 | 4210700 | 0.15 | 0.31 |
| Abercrombie & Fitch | ANF | 50.86 | 1973000 | 1.32 | 2.53 |
| Abitibi-Consolidated | ABY | 2.61 | 240600 | 0.00 | 0.00 |
| ABM Industries Inc. | ABM | 16.44 | 102600 | 0.36 | 2.14 |
| ABN AMRO Holding N.V. | ABN | 27.47 | 195900 | 0.31 | 1.14 |

Input file 2, "buys.csv"

| |
|----------------------------------|
| Shares,Symbol,Client |
| "1000","DIS","Bill Gates" |
| "950","EDS","Ben Graham" |
| "25000","WMT","Warren Buffet" |
| "3250","AMR","Jeff Bezos" |
| "775","TSG","Wendi Deng" |
| "400","HBC","Stephen Covey" |
| "2100","HIG","Richard Branson" |
| "950","TEM","Sergey Brin" |
| "1500","AGE","Michael Bloomberg" |
| "5000","BAC","Donald Trump" |
| "3333","PRU","Steve Wynn" |
| "2000","ABN","Jack Welch" |
| "8500","RNT","George Soros" |
| "1000","MCK","Kerry Packer" |
| "4300","UNH","Rupert Murdoch" |
| "9000","SDS","Jesse Livermore" |
| "3500","SNE","Alan Greenspan" |
| "825","ABT","Lakshmi Mittal" |
| "9000","ABY","Robert Kiyosaki" |
| "855","ADS","Lisa Mangino" |
| "50","IBM","Rick Haines" |
| "90","SUN","Amrita Thakur" |

The goal of the following SortCL job script is to order and match these files by the ticker symbol in their second columns so that a meaningful set of output reports can be created. Field-level protections could certainly have been applied if desired.


```

### Job Controls Phase ###
/WARNINGSON
/MONITOR=4

### Input Phase ###
/INFILE=nyse-a                                # 1st Input, tab-delimited
/FIELD=(Issue,POSITION=1,SEPARATOR='\t')
/FIELD=(Symbol,POSITION=2,SIZE=3,SEPARATOR='\t')
/FIELD=(LastTrade,POSITION=3,SIZE=5.2,SEPARATOR='\t',NUMERIC)
/FIELD=(Volume,POSITION=4,SEPARATOR='\t',NUMERIC)
/FIELD=(Change,POSITION=5,SIZE=4.2,SEPARATOR='\t', NUMERIC)
/FIELD=(Percent,POSITION=6,SIZE=4.2,SEPARATOR='\t', NUMERIC)

/INFILE=buys.csv                               # 2nd Input, CSV format
/ALIAS=buys
/INSKIP=1          # skip header record
/FIELD=(Shares,POSITION=1,SEPARATOR=',',FRAME='')
/FIELD=(Symbol,POSITION=2,SEPARATOR=',',FRAME='')
/FIELD=(Client,POSITION=3,SEPARATOR=',',FRAME='')

### Action Phase ###
/JOIN FULL_OUTER NOT_SORTED nyse-a NOT_SORTED buys \
    WHERE nyse-a.Symbol EQ buys.Symbol

#### Output File 1 - 2D BI Report with Selection ####
/OUTFILE=TradingA                             # Summary record format
/HEADREC="                                     -----\n"
/FIELD=(New_balance,POSITION=50,SIZE=14.2,currency)
/SUM New_balance from (nyse-a.LastTrade * buys.Shares) # Expression
logic

/OUTFILE=TradingA                             # Detail record format
/HEADREC="Client      Symbol      Shares LastTrade  Shares*LT      Ln.\n\n"
/FIELD=(buys.Client,POSITION=1,SIZE=17)
/FIELD=(buys.Symbol,POSITION=20,SIZE=5)
/FIELD=(nyse-a.Symbol,POSITION=28,SIZE=5)
/FIELD=(buys.Shares,POSITION=35,SIZE=5)
/FIELD=(nyse-a.LastTrade,POSITION=45,SIZE=5.2,NUMERIC)
/FIELD=(product,POSITION=54,NUMERIC, IF nyse-a.Symbol NE buys.Symbol      \
    THEN "" ELSE nyse-a.LastTrade * buys.Shares)
/FIELD=(Sequencer,POSITION=66,SIZE=4)          # Creates sliding index column

#### Output File 2 - Data Interchange Format ####
/OUTFILE=TradingA.xml
/PROCESS=XML
/FIELD=(Client,POSITION=1,SEPARATOR='|',XDEF="/Trades/Buy@Client")
/FIELD=(Symbol,POSITION=2,SEPARATOR='|',XDEF="/Trades/Buy/Symbol")
/FIELD=(Shares,POSITION=3,SEPARATOR='|',XDEF="/Trades/Buy/Shares")
/INCLUDE WHERE nyse-a.LastTrade GT 10 AND buys.Shares GT 0

```

Output File 3 - Web-ready Summary Report

```

/OUTFILE=TradingA.htm                                # Summary record format
/DATE="</FONT><B></TD>\n<TD align=right><B><U><FONT SIZE=+2>          \
          <FONT COLOR='GREEN'>"
/FIELD=(New_balance,SIZE=15,CURRENCY)                # Derived in prior output spec!
/DATE="</FONT></U></B></TD>\n</TR>\n"
/SUM New_balance from (nyse-a.LastTrade * buys.Shares) WHERE symbol NE "RNT"
/FOOTREC="</TABLE><BR>\nCreated on </B>%s.\
          <HR></BODY>\n</HTML>","AMERICAN_DATE

/OUTFILE=TradingA.htm                                # Detail record format
/HEADREC="<HTML><HEAD>\n<TITLE>HTML produced by SORTCL \
          </TITLE>\n</HEAD>\n<BODY><H2><FONT COLOR='RED'>Trading Summary \
          </H2>\n<TABLE CELLPADDING=4 CELLSPACING=1 BORDER COLS=5>\n"
/OMIT WHERE Symbol EQ "RNT" OR SYMBOL EQ "" # Selection applied in display
/DATE="<TR>\n<TD><I>"
/FIELD=(buys.Client)
/DATE="</B></TD>\n<TD align=right><FONT COLOR='BLUE'>"
/FIELD=(buys.Symbol)
/DATE="</TD>\n</TR>\n"

```

The following command ran the job above, producing all targets and job statistics at once:

```
sortcl /SPECIFICATION=stockjoin.scl
```

And as the outputs were created, requested warning and monitor messages were displayed:

```
+16 stockjoin.scl: warning (101): "PRECISION" ambiguous reference
+18 stockjoin.scl: warning (101): "PRECISION" ambiguous reference
+19 stockjoin.scl: warning (101): "PRECISION" ambiguous reference
+38 stockjoin.scl: warning (101): "PRECISION" ambiguous reference
+47 stockjoin.scl: warning (101): "PRECISION" ambiguous reference
    warning TradingA
    gap [1 -> 49]
    warning TradingA
    gap [18 -> 19]
    gap [25 -> 27]
    gap [33 -> 34]
    gap [40 -> 44]
    warning TradingA.htm
        overlap [1 -> 14]
    warning TradingA.xml
        missing field before field 175
        missing field before field 177

CoSort Version 9.5.3 D90070608-1306 Copyright 1978-2014 IRI, Inc. www.iri.com
EDT 05:25:53 PM Monday, April 14 2014. #95162.9543 2 CPUs
Expires Dec 31, 2012      Monitor Level 4
<00:00:00.00> event (57): /spec=stockjoin.scl initiated
<00:00:00.00> event (57): /infile=nyse-a initiated
<00:00:00.03> 2 CPUs
<00:00:00.14> event (59): P5a5b88 infile opened
<00:00:00.14> event (59): P5a5c08 infile opened
<00:00:00.00> event (57): /infile=buys.csv initiated
<00:00:00.06> event (66): cosort() process begins
<00:00:00.14> event (59): d:\CS_JOIN_7d8.6fc infile opened
<00:00:00.14> event (66): cosort() process begins
<00:00:00.22> event (67): cosort() process ends
<00:00:00.22> event (58): /infile=buys.csv completed
<00:00:00.33> event (61): TradingA outfile opened
<00:00:00.33> event (61): TradingA.xml outfile opened
<00:00:00.33> event (61): TradingA.htm outfile opened
<00:00:00.30> event (67): cosort() process ends
<00:00:00.30> event (58): /infile=nyse-a completed
<00:00:00.34> event (60): P5a5b88 infile closed0
<00:00:00.34> event (60): P5a5c08 infile closed0
<00:00:00.34> event (68): left 0 right 0
<00:00:00.34> event (62): TradingA outfile closed
<00:00:00.34> event (62): TradingA outfile closed
<00:00:00.34> event (62): TradingA.xml outfile closed
<00:00:00.34> event (62): TradingA.htm outfile closed
<00:00:00.34> event (62): TradingA.htm outfile closed
<00:00:00.39> event (58): /spec=stockjoin.scl completed
EDT 05:25:53      CoSort Serial # 95162.TEST 2 CPUs Expires Dec 31, 2014
```

In **TradingA**, both matches and non-matches are shown in order by ticker symbol. The cross and down-row calculations across support business intelligence and billing operations.

Output file 1, "TradingA"

| Client | Symbol | Shares | LastTrade | Shares*LT | Ln. |
|-------------------|--------|--------|-----------|--------------|-----|
| | ABB | | 12.55 | | 1 |
| | ABM | | 16.44 | | 2 |
| | ABN | | 27.47 | | 3 |
| Lakshmi Mittal | ABT | 825 | 47.25 | 38981.25 | 4 |
| Robert Kiyosaki | ABY | 9000 | 2.61 | 23490.00 | 5 |
| Lisa Mangino | ADS | 855 | | | 6 |
| Michael Bloomberg | AGE | 1500 | 52.81 | 79215.00 | 7 |
| | AIC | | 4.84 | | 8 |
| Jeff Bezos | AMR | 3250 | | | 9 |
| | ANF | | 50.86 | | 10 |
| | AOS | | 42.40 | | 11 |
| Donald Trump | BAC | 5000 | | | 12 |
| Bill Gates | DIS | 1000 | | | 13 |
| Ben Graham | EDS | 950 | | | 14 |
| | GFW | | 25.05 | | 15 |
| | GFZ | | 24.71 | | 16 |
| Stephen Covey | HBC | 400 | | | 17 |
| Richard Branson | HIG | 2100 | | | 18 |
| Rick Haines | IBM | 50 | | | 19 |
| Kerry Packer | MCK | 1000 | | | 20 |
| Steve Wynn | PRU | 3333 | | | 21 |
| George Soros | RNT | 8500 | 24.05 | 204425.00 | 22 |
| Jesse Livermore | SDS | 9000 | | | 23 |
| Alan Greenspan | SNE | 3500 | | | 24 |
| Amrita Thakur | SUN | 90 | | | 25 |
| | SXA | | 25.00 | | 26 |
| Sergey Brin | TEM | 950 | | | 27 |
| | TNT | | 103.0 | | 28 |
| Wendi Deng | TSG | 775 | | | 29 |
| Rupert Murdoch | UNH | 4300 | | | 30 |
| Warren Buffet | WMT | 25000 | | | 31 |
| | | | | ----- | |
| | | | | \$346,111.25 | |

The next target, in valid XML format, contains only the selected fields and condition results:

Output file 2, "TradingA.xml"

```

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
= <Trades>
  = <Buy Client="Lakshmi Mittal">
    <Symbol>ABT</Symbol>
    <Shares>825</Shares>
  </Buy>
  = <Buy Client="Michael Bloomberg">
    <Symbol>AGE</Symbol>
    <Shares>1500</Shares>
  </Buy>
  = <Buy Client="George Soros">
    <Symbol>RNT</Symbol>
    <Shares>8500</Shares>
  </Buy>
</Trades>

```

The last output target is formatted with HTML and thus ready for web posting. Notice how the omission of the RNT transaction changes the grand total from the first output file, and how SortCL supports the use of 'conversion specifiers' to include system information (in this case, the date below) in its reports:

Output file 3, "TradingA.htm"

Trading Summary

| | |
|--------------------------|-----|
| <i>Lakshmi Mittal</i> | ABT |
| <i>Robert Kiyosaki</i> | ABY |
| <i>Lisa Mangino</i> | ADS |
| <i>Michael Bloomberg</i> | AGE |
| <i>Jeff Bezos</i> | AMR |
| <i>Donald Trump</i> | BAC |
| <i>Bill Gates</i> | DIS |
| <i>Ben Graham</i> | EDS |
| <i>Stephen Covey</i> | HBC |
| <i>Richard Branson</i> | HIG |
| <i>Rick Haines</i> | IBM |
| <i>Kerry Packer</i> | MCK |
| <i>Steve Wynn</i> | PRU |
| <i>Jesse Livermore</i> | SDS |
| <i>Alan Greenspan</i> | SNE |
| <i>Amrita Thakur</i> | SUN |
| <i>Sergey Brin</i> | TEM |
| <i>Wendi Deng</i> | TSG |
| <i>Rupert Murdoch</i> | UNH |
| <i>Warren Buffet</i> | WMT |
| \$141,686.25 | |

Created on April 14, 2014.

IRI WORKBENCH

CoSort 9.5 includes a Graphical User Interface (GUI) for the Sort Control Language (SortCL) program. IRI Workbench is a plug-in to the Eclipse Integrated Development Environment (IDE) that helps users create, maintain, and execute SortCL and other IRI software jobs. The Workbench provides new functional and ergonomic enhancements, including:

New job wizards, form dialogs and syntax-aware script editing

IRI Workbench utilizes several different job presentation facilities within Eclipse to improve the SortCL job design experience. The wizards take you from source and metadata specification through the action phase of a job, and finally to the specification of one or more targets and formats. Wizard and form dialog results help populate and modify SortCL job scripts. A syntax-aware editor also facilitates valid script creation and modification. Job changes made in either the editor or the GUI dynamically update the script.

The top window in the figure below shows the editor view of a join job script. You can invoke the join wizard (shown in the bottom left) from the main menu and within the script editor view. The join wizard allows you to specify all details of a multi-table join action. The bottom right window shows the target field layout of this job script. Note that multiple input sources (shown in upper tabs) and output targets (shown in lower tabs) are presented in this view to enable you to specify file and table target layouts quickly and efficiently.

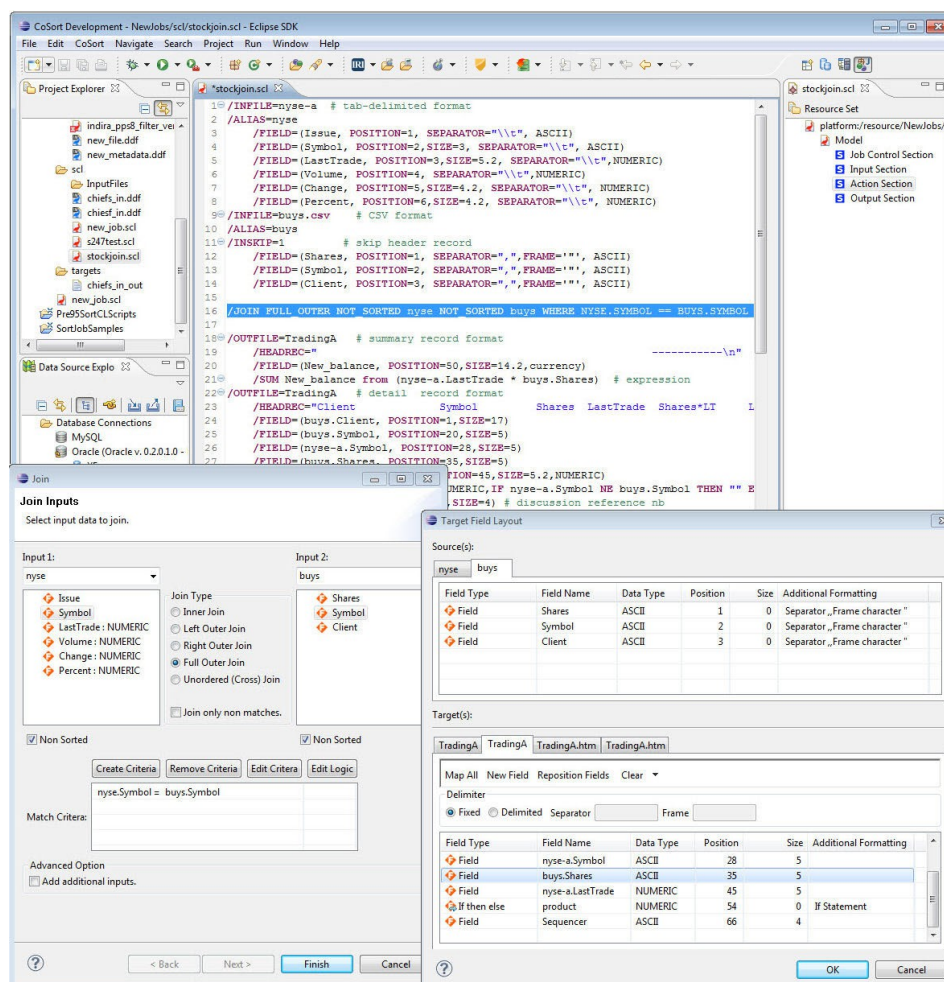


Figure 3 IRI Workbench Overview

ODBC-connected (DB table) data sources and targets

In addition to their flat file inputs, CoSort users can integrate, stage, transform, protect, and report against data stored in Oracle, DB2, SQL Server, SAP, MySQL, Sybase, Excel, and other ODBC-connected sources. The Workbench uses the JDBC-ODBC bridge in the Eclipse Data Tools Platform (DTP) for viewing and selecting table data.

In the graphic below, the left panel shows the DTP view of a database. The middle panel displays the contents of the selected table, and the right panel is the Data Definition File (DDF) form editor that is used to modify the table metadata specifications for a SortCL job.

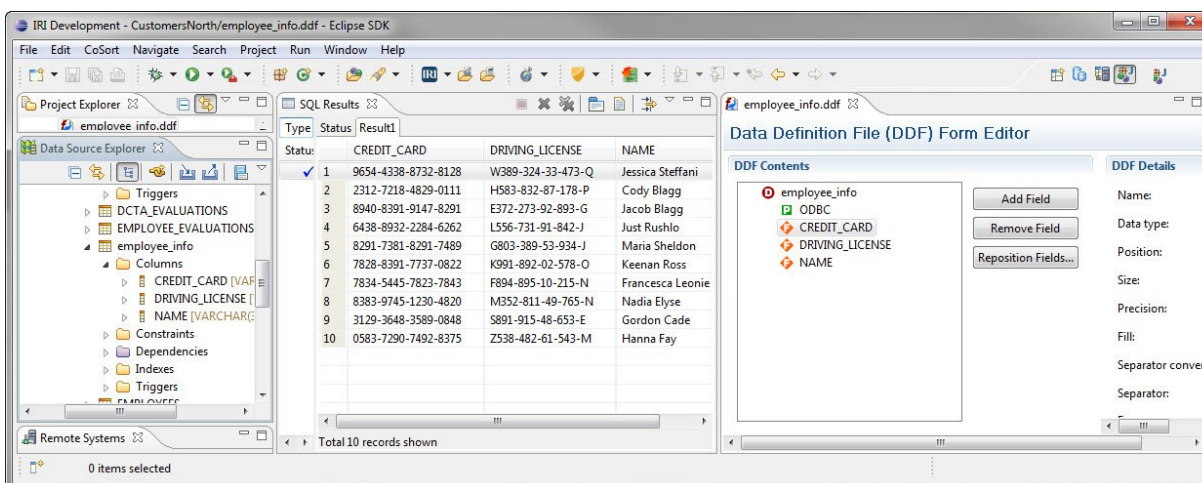


Figure 4 DTP and DDF Editor

Metadata discovery

Those familiar with CoSort know that you must define the structure of all input and output files in SortCL DDF syntax. This has traditionally been a manual field-by-field editing process, or the result of a command line conversion program like cob2ddf (for COBOL copybooks). When pre-existing metadata is not available, IRI Workbench helps users to visually define their field layouts and populate file and table metadata for use in SortCL jobs.

The following screenshot shows how users can define fixed-position fields by moving sliders to the start and end of each field in a source data preview window. Once you add a field, the bottom spreadsheet view reflects the DDF specifications which you can then modify. For records with delimited fields, a column-based editor is provided instead, as sliders are unnecessary. A HEX view option facilitates the definition of binary data.

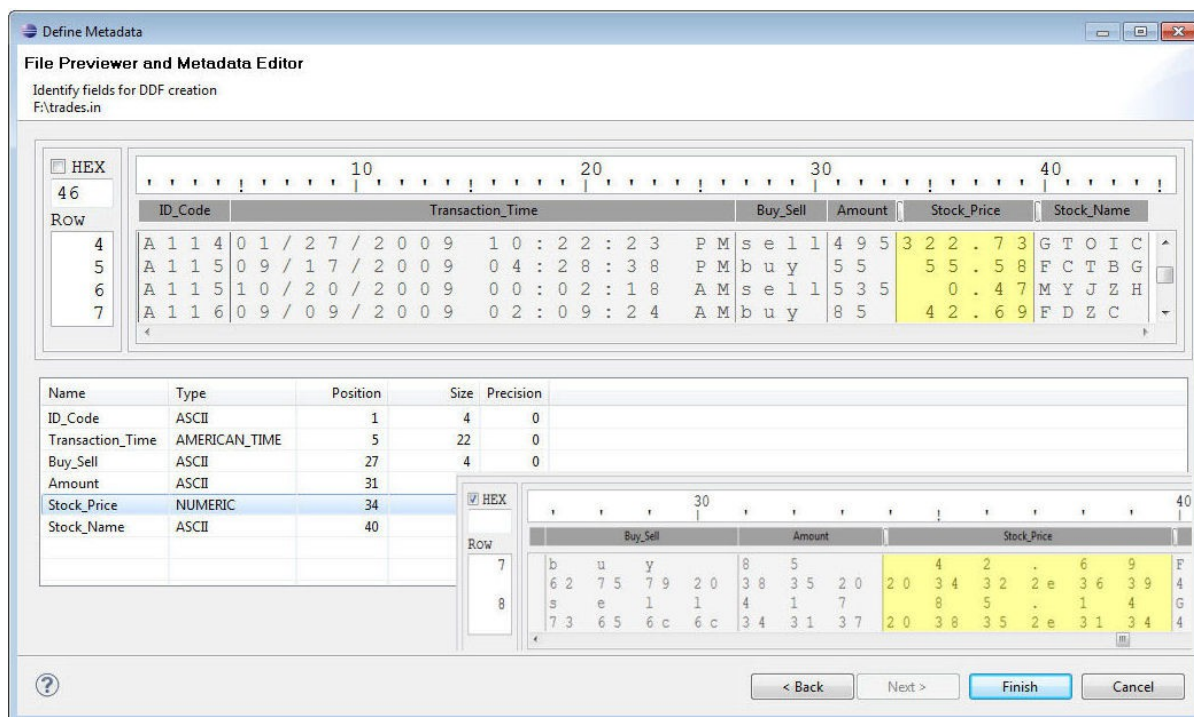
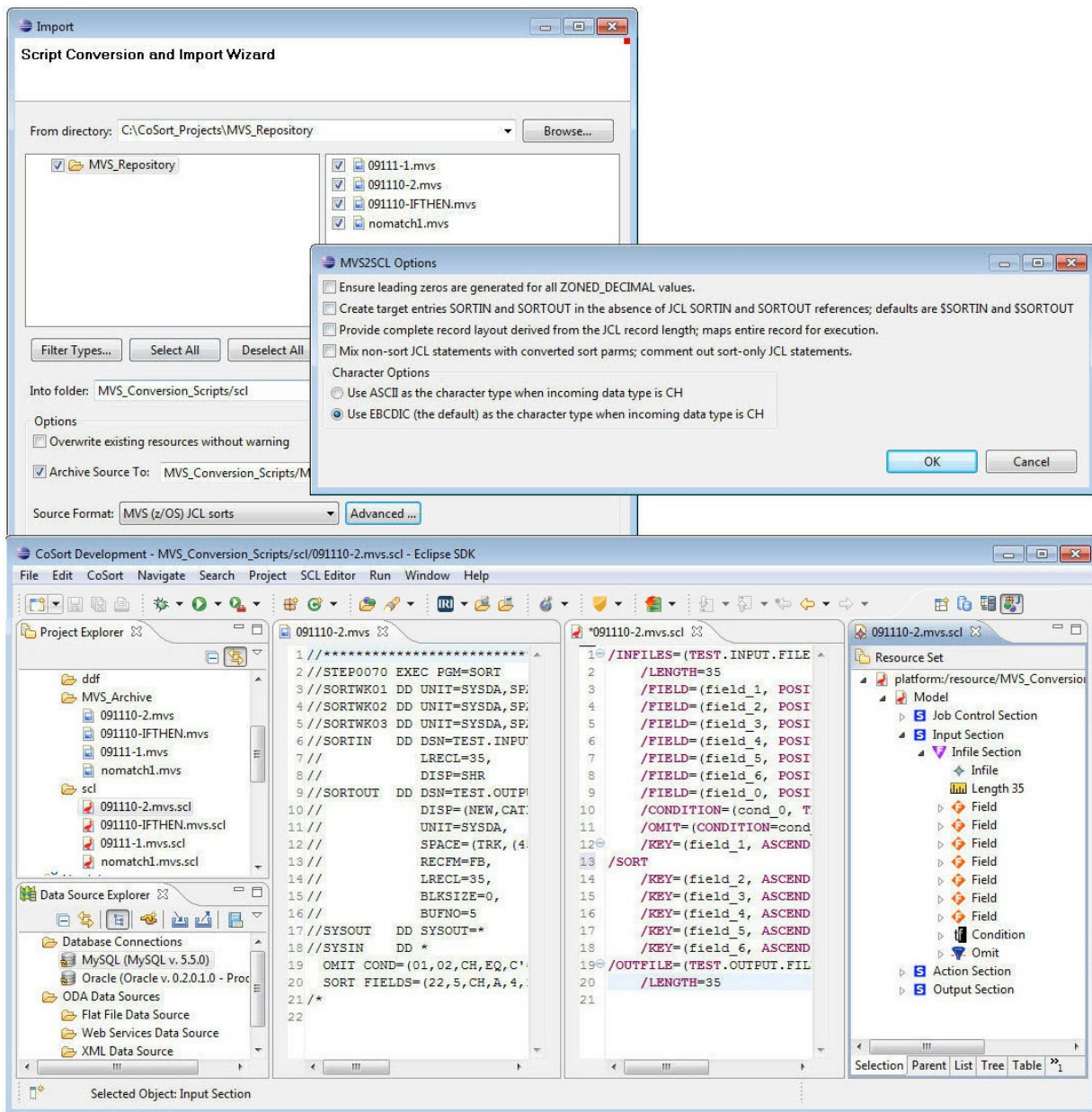


Figure 5 Define Metadata Wizard

Metadata conversion

The process of bulk metadata and third-party sort script migration to CoSort's SortCL syntax has been modernized and automated in the Workbench. Wizards exist to translate third-party data layouts into SortCL DDFs and SortCL job scripts. An example of the former would be bulk COBOL copybook conversions to SortCL .ddf targets. There is a wizard to convert one or more metadata repositories or file headers into DDF. There is also a wizard to convert and import third-party sort specifications into SortCL job scripts.

The conversion wizard shown below demonstrates the conversion of one or more JCL sort decks to SortCL job scripts. You can browse to the location of the parms, and then automatically convert them for use in SortCL. The middle window shows advanced options available for these conversions. The bottom window shows from left to right: project explorer files (including the source and target job scripts), an editor displaying the source job script, a display of the target job script, and a tree view of the components of the target job script.



SORTCL COMMAND SET

The following is a list of the core commands available in SortCL as of CoSort Version 9.5:

| | | |
|-----------------|----------------|----------------|
| /ALIAS | /INCOLLECT | /OUTFILE |
| /APPEND | /INFILE | /OUTPROCEDURE |
| /ALTSEQ | /INFILES | /OUTSKIP |
| /AVERAGE | /INPROCEDURE | /PROCESS |
| /CHARSET | /INREC | /QUERY |
| /CHECK | /INSKIP | /RC |
| /CONDITION | /JOBBCOLLECT | /RECSPPERPAGE |
| /COUNT | /JOBSKIP | /REPORT |
| /CREATE | /JOIN | /ROUNDING |
| /DATA | /KEY | /SORT |
| /DEBUG | /KEYPROCEDURE | /SPECIFICATION |
| /DUPLICATESONLY | /LENGTH | /STABLE |
| /ENDIAN | /LIBRARY | /STATISTICS |
| /EXECUTE | /LOCALE | /SUM |
| /FIELD | /MAXIMUM | /TAILREAD |
| /FILE | /MINIMUM | /TAILSKIP |
| /FOOTREC | /MEMORY-WORK | /TAILWRITE |
| /HEADREAD | /MERGE | /TEMPLATE |
| /HEADREC | /MONITOR | /UPDATE |
| /HEADSKIP | /NODUPPLICATES | /WARNINGSOFF |
| /HEADWRITE | /OMIT | /WARNINGSON |
| /INCLUDE | /OUTCOLLECT | |

Note that each command may contain additional parameters that expand its functionality. For example, the /JOIN ONLY command will eliminate the inner join results of a full outer join. Note also that many external functions can be invoked beyond the command set. For example, the encryption and other de-identification functions, as well as set file lookups, are specified as functions within /FIELD statements.

METADATA CONVERSION TOOLS

CoSort Version 9.5 includes several tools to translate existing flat file layout (and sort job) metadata for use in CoSort's Sort Control Language (SortCL) program. IRI has also partnered with a leading metadata conversion company, Meta Integration Technology, Inc. (MITI) to automatically create SortCL-compatible data definition file layouts from third-party BI, ETL and relational metadata.

About SortCL Metadata

SortCL's explicit text-based metadata is straightforward and self-documenting -- making it easy to learn, use, and audit. You can store your files' field (column) layouts in reusable Data Definition File (.ddf) repositories. You can paste or reference these definitions in your SortCL job script. SortCL uses /FIELD= statements to identify column:

- Names or Aliases
- Sizes or Ranges
- Positions or Delimiters
- Data Types
- Conditions and Expressions
- Security and Other Functions

Third-Party Metadata

File layout translators included in the CoSort package can migrate your existing field/column descriptions into SortCL (and RowGen) .ddf repositories. These tools can reduce or eliminate the overhead associated with migrating metadata to a .ddf from:

- COBOL copybooks (cob2ddf)
- comma-separated values (csv2ddf)
- LDIF (ldif2ddf)
- XML (xml2ddf)
- W3C extended log format (elf2ddf)
- Oracle SQL*Loader control files (ctl2ddf)
- ODBC table data

Therefore, if you already have file layouts defined in the above applications, you can automatically reproduce that metadata for use in any SortCL manipulations. If your file layouts exist in *other* formats (including CWM, DSX, UML, XMI and XML), the Meta Integration Model Bridge (MIMB) will convert these repositories into SortCL .ddf syntax. This precludes the need for manually re-defining flat file field layouts for reference in SortCL applications.

Sort Program Conversions

The included mvs2scl, and vse2scl utilities convert legacy MVS JCL, and VSE JCL, respectively, to functionally-equivalent SortCL job specifications. Conversion tools cannot translate every script. However, field re-casting usually works, and manual translation of source scripts to SortCL equivalents is usually not difficult. For more information on, and examples of, these tools, see IRI's Legacy Sort Migration white paper.

Another CoSort tool -- called sorti2scl -- translates CoSort's sort interactive (SortI) program job specifications into their SortCL job equivalents. See the SORTI PROGRAM chapter.

UNIX SORT REPLACEMENT (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 a 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 the following data sample:

Input File 1, "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 field, use the command:

```
/path2/cosort95/bin/sort -k 2 chicago -o out1
```

Output File, "out1"

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

To sort starting at the second character of the first field, use the following command:

```
sort -k 1.2 chicago -o out2
```


Output File, "out2"

```
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 Beginnings 8.50 Prentice-Hall
4877 Going Nowhere 17.95 Valley Kill
3928 Not On Call 9.99 Harper-Row
```

The CoSort /bin/sort replacement also supports legacy Unix sort options. For example:

```
sort -t, +3 +2nr -3 chicago -o out3
```

Output File, "out3"

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

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]** flags are supported indirectly via values defined in your cosortrc file. For more information, see the SYSTEM TUNING chapter.

SORT INTERACTIVE PROGRAM (SORTI)

Each CoSort package also includes a menu-driven prompting program for users who may not have prior experience with data sorting software, or with prior metadata. The purpose of CoSort's "Sort Interactive" or SortI program is to allow users to easily create and run sort/merge specifications from any Unix, Linux or Windows command line.

CoSort's interactive and batch **SortI** program is invoked with the command:

```
sorti
```

A banner appears similar to the one below:

```
C:\iri\CoSort95\bin>sorti

CoSort Sort Interactive Version 9.5  D90110608-1306
Copyright 2014 Innovative Routines International, Inc.
S/N 95162.9543  2 CPUs
Eastern Daylight Time 04:15:51 PM Monday, April 14 2014

Directory: C:\iri\CoSort95\bin
# Threads: 2
BlockSize: 2097152  bytes
MaxMemory: 268435456 bytes
WorkAreas: d:\temp\sortwork, e:\temp2

Responses: <cr> takes offered default
           'stop' restarts the Action
           'help' explains each query
           $variable defined in shell
           # starts comment for batch
           !command for shell command

Action:  SORT  MERGE  DISPLAY  GUIDE (default)  END:
```

The SortI user is then prompted for responses. For example:

```

Action:   SORT   MERGE   DISPLAY   GUIDE (default)   END: SORT

INPUT   Record length 0 (variable (default)) / fixed #: 0
        Number of files: 0 (proc) / +- numb (1 default): 1
        Input file 1:  sales2.dat

KEYS [number] [stable] [unique / dus_only] (1 default): 1
  Key 1   Direction: ASCENDING (default) or DESCENDING: ASC
          Location: FIXED (default) / <separator char>: FIXED
          Starting byte position: 1 (default): 22
          Field Size: 32767 (maximum) 50 (default): 10
          Form: ALPHA (def), NUM, DATE, TIME, TIMESTAMP: DATE
          Type AMERICAN (default) or LIST to show options: LIST

          TYPE                DATE                TIME
          -----
          AMERICAN             (MM/DD/YYYY Hr:Mn:Ss AM)
          EUROPEAN              (DD.MM.YYYY Hr.Mn.SS)
          JAPANESE              (YYYY-MM-DD Hr:Mn:Ss)
          ISO                   (YYYY-MM-DD Hr.Mn.Ss)
          CS_Y2K_ASCII_JULIAN (YYDDD)
          CS_Y2K_ASCII_YR      (YY)

```

Users are also prompted to save their job specifications to a parameter file that can be run in batch mode. In this case, specifications were saved to *sales2_sort.spc*. To run the same job in a batch stream, the shell command would be:

```
sorti sales2_sort.spc
```

To expand beyond sorting the input file, and to leverage the extensive data transformation, reformatting, and reporting capabilities available through CoSort's SortCL program, a "SortI – to – SortCL" parameter translation tool is also provided in the CoSort package. Running the tool **sorti2scl** against the original SortI parameters will create the equivalent file layout and manipulation metadata for SortCL. For example, this command:

```
sorti2scl sales2_sort.spc sales2_sort.scl
```

will automatically build this SortCL job script equivalent:

```

# sortcl specs created by sorti2scl on Mon April 18 10:15:45
# sorti specifications by cosort_user (user)
# Tue April 19 09:31:12 2012
/INFILE=sales2.dat
/SORT
  /KEY=(POSITION=22,SIZE=10,ASCENDING,ISO_DATE)
/OUTFILE=sales2_sorted

```

COBOL MIGRATION TOOLS

CoSort ships with the tools, libraries, and documented examples you need to achieve a wide range of COBOL data migration and processing goals on open systems:

Sorting and Migrating COBOL Data

All CoSort interfaces support MF and RM COBOL data type collation while SortCL also handles conversion. CoSort includes a COBOL copybook metadata translation tool to leverage your file layouts in SortCL applications, and JCL sort parm conversion tools to leverage your MVS and VSE cards. CoSort can also sort EBCDIC data and can sort ASCII data in EBCDIC order. Multiple conversions can be done while simultaneously sorting.

Accelerating Native Sort Calls

CoSort is faster than your compiler's built-in sort function, and CoSort packages include several tools and methods to improve COBOL sort performance:

- Sort replacement for ACUCOBOL-GT
- Sort replacement for Micro Focus COBOL
- Serial and concurrent system calls to SortCL
- Static and dynamic API calls to CoSort libraries

Sorting and Converting Index, Variable Length & Blocked Files

CoSort's SortCL tool can perform multiple manipulations and conversions on ACUCOBOL-GT Vision files, Micro Focus variable length records, and index files in MF-ISAM and VISION format. In the SortCL (4GL) job script, you define your input and output file formats and record layouts, along with your data filtering and transformation (sort, join, aggregate, etc.). You can integrate these formats with files in other formats all at the same time, and write output files in the same or different file format.

Generating Reports

CoSort's SortCL program includes a wide range of reporting features you can exploit to customize detail and summary targets for presentation and hand-offs to other tools.

Protecting Sensitive Data

Either as a separate process, or in combination with the above SortCL activities, you can also engage field-level protection functions like encryption and de-identification.

Creating Safe COBOL Test Data

CoSort's Test Data (RowGen) product uses the same syntax as SortCL to define the layout of COBOL index and sequential files containing randomly-generated or individually-selected test data.

Auditing Data and Applications

For data governance and other tracking purposes, SortCL also offers several logging options, including a query-ready XML file with complete scripts (containing field privacy protection specifications).

APPLICATION PROGRAMMING INTERFACES (APIs)

In addition to the many standalone utilities and third-party sort replacements in each CoSort package, there are two high-performance, thread-safe sorting libraries you can call into your software. Each API satisfies a different class of requirements.

You can link these C routines statically or dynamically, and the same calling code runs across all Unix, Linux and Windows platforms. Both libraries leverage the same multi-threaded CoSort sorting routine against any volume of input. Inputs and outputs can be in the form of files, pipes, records and record buffers (blocks) streaming to and from multiple calls simultaneously from your applications.

Integrating Sort/Merge Operations Only

The traditional CoSort API is now thread-safe, and is documented as `cosort_r()`. You can call `cosort_r()` to speed operations that sort or merge high volumes of data. Because your programs configure the input, compare, and output processes into the CoSort engine, you also can apply your own selection and comparison criteria.

The 'r' in `cosort_r()` refers to the reentrant nature of the call; you can call the function recursively from multiple processes. This means you can specify multiple sort orderings on the same input, and in the same pass. Flexible architecture also allows you to manage several sort jobs from within a single process, and from within as many processes as you like.

Integrating Multiple Transformation Functions

CoSort's multi-purpose SortCL tool is also available for thread-safe application calls and can enable the simpler execution of scripts. Embed CoSort's `sortcl_routine()` library to speed and combine many functions, including:

- Sort/Merge
- Match/Join
- Aggregate/Calculate
- Filter/Scrub
- Type-Conversions
- Encrypt/De-ID
- Reporting
- Random Data Generation

This API gives you access to all the data transformation, business intelligence, protection, and prototyping functions available in SortCL's data definition and manipulation syntax. Integrating `sortcl_routine()` into an ETL environment allows you to source and target database tables, as well as files, pipes, and custom input/output procedures.

The following C program demonstrates the simplicity of calling SortCL scripts via the `sortcl_routine(api)`:

```

/* **** */
* Copyright 1978-2014 CoSort / Innovative Routines International (IRI), Inc.
* All Rights Reserved.
*
* SortCL API example to run a script file
* ScriptFile: keyproc.scl
* Input File: chiefs
* Output File: chiefs.out
* **** */
/* cosort header file */

#if defined (_WIN32)
#define __DLL_IMPORT__ /* IMPORTANT: needed to work with Windows DLL */
#pragma pack(1) /* or compile with /Zp1 option */
#endif /* (_WIN32) */
#include "cosort.h"
#include "sortcl_routine.h"

int main()
{
int iRetVal; /* return value */
cs_sortcl_t* sortcl; /* main sortcl context variable */
/* allocate the main sortcl context variable */
sortcl = sortcl_alloc();
if (sortcl) {
/* call sortcl_routine api */
iRetVal = sortcl_routine(sortcl, "/spec=keyproc.scl");
/* free the sortcl variable */
sortcl_free(sortcl);
}
/* if return value != 0 it is an error */
return (iRetVal);
}

```

You can make calls to the sortcl_routine library from any language that links to C. For more information about the content of SortCL job specification (.scl) files, see the functional description and scripting samples in the SORTCL PROGRAM chapter.

COBOL programmers may also wish to consider the ability to call SortCL scripts as system calls; for example:

```
CALL "system" USING "sortcl /SPECIFICATION=keyproc.scl &"
```

where **keyproc.scl** starts in the background while other COBOL program functions run.

SYSTEM TUNING

High-volume sorting and related data manipulations can be very resource-intensive. CoSort can achieve scalability that is nearly linear through its proprietary processing techniques and the proper application of system tuning. At the same time, however, CoSort is designed to be a good neighbor in a multi-user computing mix.

System administrators can combine kernel and CoSort tuning to optimize the performance of large data transformations without unduly impacting concurrent jobs. To prioritize CoSort operations at the system (global), user, or job level, adjust the values associated with the parameters shown below in your resource control file(s) (on Unix) or Windows registry:

Threads

On SMP platforms, CoSort users can manually set the number of sub-processes that sorts and related transformations will create. In most cases, the closer this value is set to the number of actual processors and disk drives on the system, the better the performance.

Memory Allocation

By assigning, or limiting the amount of random access memory available for sorting, system administrators can maximize the efficiency of jobs according to their desired system priority. Like CoSort's other system tuning controls, this variable allows users to determine how much impact CoSort will have on concurrently running applications.

Block Size

Blocks of memory are used as buffers to hold data temporarily. The data move through input, sort and output phases of the CoSort utilities (or a customized front-end), and between an application (calling) program and the cosort() engine. The size of these blocks determines how often disk I/O will be performed, and is a function of how much memory is available.

Overflow Storage

Overflow occurs in sorting when there are more input records than can be held in memory. When all the overflow data are distributed to temporary files, these files and the internal data are merged to produce output. Users can control the location of these temporary files and distribute them across multiple file systems. Where multiple drives and threads are specified, speed can increase in the later stages of the sort as the data is read back (into the output file/s) in parallel. You can also use input selection to reduce processing volume and temporary space requirements at runtime.

CoSort resource controls also include these application-specific values:

Record Terminator

Where variable length output is specified, any record terminating character can be specified. The default option is to use the same terminator present in the input data source, if applicable (which is typically a linefeed on Unix and carriage-return/linefeed on Windows). Alternatively, you can choose either of these styles, regardless of the terminator type used on input -- or you can select your own special terminator character(s), including a NULL character ("").

Century Window

Users with non-Y2K-compliant (2-digit date) data can specify the minimum year for CoSort to sort after 1999. This "sliding" feature allows custom collation for any century-bordering dates.

Pause / Resume

During large sort operations that require the creation of temporary work files, CoSort can warn users when temporary disk space is exhausted, and allow for ad hoc re-allocation so that the job can continue without having to be re-started.

Runtime Monitoring

By specifying various levels of verbosity, CoSort users can view on-screen job progress reports with timestamps and event messaging. Events include the opening and closing of input, work, and output files and the number of records accepted, rejected or processed. Displays range from off (level 0) to showing every single record (number) being processed (level 9).

Execution Log

Basic runtime information can be directed to one or more files to archive CoSort jobs and their performance. This log file is a self-appending text file. A self-replacing text file called *.cserrlog* is created during each run to record tuning and version information, and any error messages, in the event of an abnormal termination (for debugging purposes).

Audit Log

Detailed runtime information can be optionally directed to a self-appending, query-ready XML log file that contains environmental and performance details, and the entire SortCL job script, to record every aspect of the application (data definitions, manipulations and protections) in order to verify compliance with procedures and data privacy regulations.

A complete description of all parameters, and how to tune them, can be found in Appendix D of the **CoSort User Manual**.

TECHNICAL SPECIFICATIONS

CoSort is general-purpose data manipulation package for big data transformation, reporting, protection and conversion. All of its native utilities, third-party sort replacements, and API calls link to the same coroutine sort engine which processes all file formats and record types, and provides in-memory, record-level processing for file and data manipulation. CoSort uses these techniques, along with granular tuning parameters, to both optimize application throughput and prioritize resources in the multi-user mix. CoSort's unique approach cuts production times without compromising performance of concurrent jobs.

Installation

- ❖ Distributed via internet or user-specified media
- ❖ Loads in under two minutes
- ❖ Menu-driven setup and configuration utility

Invocation

- ❖ Command line (including pipe sequences), shell commands and batch scripts
- ❖ IRI Workbench – Eclipse GUI for Windows (JRE required)
- ❖ Application calling programs as a standalone executable, subroutine or coroutine call, with or without additional exit routines

Ease of Use

- ❖ Processes data using record layouts and SQL-like field descriptions within applications or centralized data definition files (DDF)
- ❖ Converts and processes native COBOL copybook, Oracle SQL*Loader control file, CSV, and W3C extended log format (ELF) file layouts
- ❖ SortCL DDF is a supported MIMB metadata format
- ❖ Provides on-line help, pre-runtime application validation, and runtime errors
- ❖ Leverages centralized application and file layout definitions (metadata repositories)
- ❖ Reports problems to standard error when invoked from a program, or to an error log
- ❖ Runs silently or with verbose messaging without user intervention
- ❖ Allows user control over the amount of informational output produced
- ❖ Generates a query-ready XML audit log for data forensics and privacy compliance
- ❖ Describes commands and options through man pages and on-line documentation
- ❖ Fully supports job design, execution, modification, and tuning in familiar Eclipse GUI
- ❖ Easy-to-use interfaces and seamless third-party sort replacements preclude the need for training classes; however, advanced training is available in Florida or at user sites
- ❖ Phone, web, and email support available directly from the product developers
- ❖ Local language support is available from more than 30 international offices

Resource Control

- ❖ Sets and allows user modification of the maximum and minimum number of concurrent threads for sorting on multi-CPU and multi-core systems
- ❖ Uses a specified directory or a combination of directories for temporary work files
- ❖ Limits the amount of main and virtual memory used during sort operations
- ❖ Sets the size of the memory blocks used as physical I/O buffers

Input and Output

- ❖ Processes any number of structured data sources of any size, and any number of records, fixed or variable length (to 65,535 bytes)
- ❖ Sources can be: flat files, ODBC-connected tables or spreadsheets, an input procedure, **stdin**, a named pipe, a table in memory, or from an application program
- ❖ Supports the use of environment variables and wildcards in the specification of input and output files, as well as absolute path names and aliases
- ❖ Accepts and outputs fixed- or variable-length records with delimited fields
- ❖ Generates one or more output targets, and/or summary information, including formatted and BIRT-ready reports
- ❖ Returns sorted, merged, or joined records one (or more) at a time to an output procedure, to **stdout**, a named pipe, a table in memory, one or more new or existing flat files, or to an application program
- ❖ Outputs optional sequence numbers with each record, at any starting value, for indexed loads and/or reports
- ❖ Synthesizes randomly generated or randomly hand-selected field test data values.

Record Selection and Grouping

- ❖ Includes or omits input or output records using field-to-field or field-to-constant comparisons
- ❖ Compares on any number of data fields, using standard and alternate collating sequences
- ❖ Sorts and/or reformats groups of selected records
- ❖ Matches two or more sorted or unsorted files on inner and outer join criteria using SQL-based condition syntax
- ❖ Skips a specified number of records, bytes, or a record header
- ❖ Processes a specified number of records or bytes, including a saved header
- ❖ Eliminates or saves records with duplicate keys.

Sort Key Processing

- ❖ Allows any number of key fields to be specified in ascending or descending order
- ❖ Supports any number of fields from 1 to 65,535 bytes in length
- ❖ Orders fields in fixed position or floating (on one or more delimiters)
- ❖ Supports numeric keys, including all C, FORTRAN, and COBOL data types
- ❖ Supports single and multi-byte character keys, including ASCII, EBCDIC, ASCII in EBCDIC sequence, Thai characters, and natural (locale-dependent) values
- ❖ Supports American, European, ISO and Japanese timestamps
- ❖ Supports Unicode and double-byte characters like Big5, EUC-TW, UTF32, and S-JIS
- ❖ Allows left or right alignment and case shifting of character keys
- ❖ Accepts user compare procedures for multi-byte, encrypted and other special data
- ❖ Performs record sequence checking
- ❖ Maintains input record order (stability) on duplicate keys
- ❖ Controls treatment of null fields when specifying floating (character separated) keys
- ❖ Collates (and converts between many of) the following data types (formats):

| Form Group | Form Type |
|------------|------------|
| 0 | Alphabetic |
| 1 | Numeric |
| 2 | Date |
| 3 | Time |
| 4 | Timestamp |

| Single Byte Types of Form 0 | Reference Standard | Form 1 - Numeric Types | C Types |
|--|--|------------------------|------------------------------------|
| ASCII [LATIN1] | ISO 8859-1 | CHAR | Character, Natural (ASCII) |
| EBCDIC | IBM Standard | SCHAR | Character, Signed |
| LATIN2 | ISO 8859-2 | UCHAR | Character, Unsigned (EBCDIC) |
| LATIN3 | ISO 8859-3 | SHORT | Integer, Short Signed |
| BALTIC | ISO 8859-4 | USHORT | Integer, Short Unsigned |
| CYRILLIC | ISO 8859-5 | INT | Integer, Natural Signed |
| ARABIC | ISO 8859-6 | UINT | Integer, Natural Unsigned |
| GREEK | ISO 8859-7 | LONG | Integer, Long Signed |
| HEBREW | ISO 8859-8 | ULONG | Integer, Long Unsigned |
| TURKISH [LATIN5] | ISO 8859-9 | FLOAT | Single Precision Float |
| LATIN6 | ISO 8859-10 | DOUBLE | Float, Double Precision |
| ASC_IN_EBC | e.g., "123">"ABC" | | |
| ASC_IN_NATURAL | e.g., "ABC">"A-D" | | |
| Multi-Byte Types of Form 0 | Reference Standard | Form 1 - Numeric Types | Ryan-McFarland (Liant) COBOL Types |
| JOHAB | KS X 1001:1992 | RM_COMP | COMP, Signed |
| KEF | Korean EBCDIC | URM_COMP | COMP, Unsigned |
| EHANGUL | IBM DBCS-HOST | RM_CMP1 | COMP-1 |
| HHANGUL | Hitachi Hangul | RM_CMP3 | COMP-3, Signed |
| UTF8/UNICODE | ISO 10646:1993-1 | URM_CMP3 | COMP-3, Unsigned |
| UTF16/UCS2/UNICODE | ISO10646 | RM_CMP6 | COMP-6 |
| UTF32/UCS4/UNICODE | ISO10646 | RM_DISP | DISP, Signed |
| GBK/EUC_CN | ISO10646 (China) | URM_DISP | DISP, Unsigned |
| BIG5 | BIG5 (Hong Kong) | RM_DISPSSL | DISP, Sign Leading |
| EUC_TW | CNS 11643-1992 (Planes 1 - 3) | RM_DISPSSL | DISP, Sign Leading Separate |
| EUC_KR | KS X 1001:1992 | RM_DISPST | DISP, Sign Trailing |
| EUC_JP | JIS X 0201-1976 JIS X 0208-1990 H/W Katakana JIS X 0212-1990 | RM_DISPSTS | DISP, Sign Trailing Separate |
| SJIS (Shift_JIS) | JIS X 0208-1990 | | |
| IBM_DBCS_HOST (Mainframe Encoding) and IBM_DBCS_PC (PC Encoding) | IBM Japanese, IBM Korean, IBM Simplified Chinese, and, IBM Traditional Chinese | | |
| Form 1- Numeric Types | Alphanumeric | Form 1 - Numeric Types | Micro Focus COBOL Types (Meaning) |
| ASCNUM, NUMERIC | Integers, real numbers, and floating points | MF_COMP | COMP, Signed |
| | | UMF_COMP | COMP, Unsigned |
| | | MF_CMP3 | COMP-3, Packed Decimal |
| | | UMF_CMP3 | COMP-3, Unsigned |
| | | MF_CMP4 | COMP-4, Signed |
| | | UMF_CMP4 | COMP-4, Unsigned |
| | | MF_CMP5 | COMP-5, Signed |
| | | UMF_CMP5 | COMP-5, Unsigned |
| | | MF_CMPX | COMP-X |
| | | MF_DISP | DISP, Signed |
| | | UMF_DISP | DISP, Unsigned |
| | | MF_DISPSSL | DISP, Sign Leading |
| | | MF_DISPSSL | DISP, Sign Leading Separate |
| | | MF_DISPST | DISP, Sign Trailing |

| | | | |
|-----------------------------------|--|---|---|
| | | MF_DISPSTS | DISP, Sign Trailing Separate |
| Form 1 - Numeric Types | Miscellaneous | Forms 2-4 Date, Time and Timestamp | Syntax |
| ZONED_DECIMAL | Zoned Decimals | AMERICAN_DATE | <i>month(name or integer)/day/year</i> |
| ZONED_EBCDIC | Zoned Decimals in EBCDIC | AMERICAN_TIME | <i>hour[:minute][:second] xM</i> |
| PSIGNF | Packed Decimals | AMERICAN_TIMESTAMP | <i>month/day/year hour[:minute][:second] xM</i> |
| Form 1 - Numeric Types | EBCDIC Native RM COBOL Types | EUROPEAN_DATE | <i>day.month(name or integer).year</i> |
| ERM_COMP | COMP, Signed | EUROPEAN_TIME | <i>hour[:minute][:second]</i> |
| ERM_UCOMP | COMP, Unsigned | EUROPEAN_TIMESTAMP | <i>day.month.year hour[:minute][:second]</i> |
| ERM_CMP1 | COMP-1 | JAPANESE_DATE | <i>year-month(name or integer)-day</i> |
| ERM_CMP3 | COMP-3, Signed | JAPANESE_TIME | <i>hour[:minute][:second]</i> |
| ERM_UCMP3 | COMP-3, Unsigned | JAPANESE_TIMESTAMP | <i>Year-month-day hour[:minute][:second]</i> |
| ERM_CMP6 | COMP-6 | ISO_DATE | <i>year-month(name or integer)-day</i> |
| ERM_DISP | DISP, Signed | ISO_TIME | <i>hour[:minute][:second]</i> |
| ERM_UDISP | DISP, Unsigned | ISO_TIMESTAMP | <i>Year-month-day hour[:minute][:second]</i> |
| ERM_DISPST | DISP, Sign Leading | MONTH_DAY | Jan" < "Feb" and "Wed" < "Thu" |
| ERM_DISPSTSL | DISP, Sign Leading Separate | Year 2000 Types | Syntax |
| ERM_DISPST | DISP, Sign Trailing | Y2K_ASCII_YR | <i>2-digit year</i> |
| ERM_DISPSTS | DISP, Sign Trailing Separate | Y2K_ASCII_JULIAN | <i>5-digit Julian date</i> |
| Form 1 - Numeric Types | EBCDIC Native Micro Focus COBOL Types | ASCII Supplement | Data Example |
| EMF_COMP | COMP, Signed | ALIGNMENT NONE | " Chars " |
| EMF_UCOMP | COMP, Unsigned | ALIGNMENT LEFT | "Chars " |
| EMF_CMP3 | COMP-3, Packed Decimal | ALIGNMENT RIGHT | " Chars" |
| EMF_UCMP3 | COMP-3, Unsigned | CASEFOLD YES | " Chars " |
| EMF_CMP4 | COMP-4, Signed | CASEFOLD NO | " CHARS " |
| EMF_UCMP4 | COMP-4, Unsigned | | |
| EMF_CMP5 | COMP-5, Signed | | |
| EMF_UCMP5 | COMP-5, Unsigned | | |
| EMF_COMPMX | COMP-X | | |
| EMF_DISP | DISP, Signed | | |
| EMF_UDISP | DISP, Unsigned | | |
| EMF_DISPST | DISP, Sign Leading | | |
| EMF_DISPSTSL | DISP, Sign Leading Separate | | |
| EMF_DISPST | DISP, Sign Trailing | | |
| EMF_DISPSTS | DISP, Sign Trailing Separate | | |

| THAI_620 Single-Byte Type of Form 6 | |
|---|--|
| THAI_TIS_620 | Collation order is specified by Thai Royal Institute Dictionary 2525 B.E. Edition Encoding specified by Thai Industrial Standard 620-2533 (TIS-620) |
| Chinese Big5 Multi-Byte Types of Form 6 | |
| CHINESE_UNICODE_STROKE | Unicode 5.2.0: http://www.unicode.org/versions/Unicode5.2.0/ |
| CHINESE_BIG5,HK,MO,ROC,TW | Bihua (Stroke) |
| CHINESE_BIG5_RARE,HK_RARE, MO_RARE,ROC_RARE,TW_RARE | Windows Codepage 950: http://msdn.microsoft.com/en-us/goglobal/cc305155.aspx |
| CHINESE_BIG5_DIGITS | Encoding order |
| Chinese GBK Multi-Byte Types of Form 6 | |
| CHINESE_UNICODE_PINYIN | Unicode 5.2.0: http://www.unicode.org/versions/Unicode5.2.0/ |
| CHINESE_GBK_SIMPLIFIED, PRC_SIMPLIFIED,SG_SIMPLIFIED | Pinyin |
| CHINESE_GBK_TRADITIONAL, PRC_TRADITIONAL,SG_TRADITIONAL | Windows Codepage 936: http://msdn.microsoft.com/en-us/goglobal/cc305153.aspx |
| CHINESE_GBK_TRADITIONAL_RARE, PRC_TRADITIONAL_RARE, SG_TRADITIONAL_RARE | |
| CHINESE_GBK_DIGITS | Encoding order |
| Japanese Shift_JIS Multi-Byte Types of Form 6 | |
| JAPANESE_ALPHABET,JP_ALPHABET | Dictionary order Windows Codepage 932: http://msdn.microsoft.com/en-us/goglobal/cc305152.aspx |
| JAPANESE_HIRAGANA_BIG, JP_HIRAGANA_BIG | |
| JAPANESE_HIRAGANA_SMALL, JP_HIRAGANA_SMALL | |
| JAPANESE_HIRAGANA, JP_HIRAGANA | |
| JAPANESE_KATAKANA_FULL_BIG, JP_KATAKANA_FULL_BIG | |
| JAPANESE_KATAKANA_FULL_SMALL, JP_KATAKANA_FULL_SMALL | |
| JAPANESE_KATAKANA_FULL, JP_KATAKANA_FULL | |
| JAPANESE_KATAKANA_HALF, JP_KATAKANA_HALF | |
| JAPANESE_DIGITS,JP_DIGITS | |
| JAPANESE_UNICODE_ALPHABETIC, JP_UNICODE_ALPHABETIC | Dictionary order Unicode 5.2.0: http://www.unicode.org/versions/Unicode5.2.0/ |
| JAPANESE_UNICODE_HIRAGANA_BIG, JP_UNICODE_HIRAGANA_BIG | |
| JAPANESE_UNICODE_HIRAGANA_ SMALL,JP_UNICODE_HIRAGANA_ SMALL | |
| JAPANESE_UNICODE_HIRAGANA, JP_UNICODE_HIRAGANA | |
| JAPANESE_UNICODE_KATAKANA _FULL_BIG, JP_UNICODE_KATAKANA_FULL_BIG | |
| JAPANESE_UNICODE_KATAKANA _FULL_SMALL,JP_UNICODE_ KATAKANA_FULL_SMALL | |
| | |

| | |
|---|--|
| JAPANESE_UNICODE_KATAKANA_FULL,JP_UNICODE_KATAKANA_FULL | |
| JAPANESE_UNICODE_KATAKANA_HALF,JP_UNICODE_KATAKANA_HALF | |
| Korean KSC5601 Multi-Byte Types of Form 6 | Collation Order/ Encoding Standard |
| KOREAN_HANGUL,KR_HANGUL | Dictionary order Windows Codepage 949: http://msdn.microsoft.com/en-us/global/cc305154.aspx |
| KOREAN_HANGUL_RARE,KR_HANGUL_RARE | |
| KOREAN_DIGITS,KR_DIGITS | |
| KOREAN_UNICODE_HANGUL,KR_UNICODE_HANGUL | Dictionary order |
| Standard Unicode Data Types | Collation Order/ Encoding Standard |
| UTF16_UNICODE | Dictionary order |
| UTF16_DIGITS | Encoding order |

Record Reformatting

- ❖ Inserts, removes, resizes, and reorders fields within records
- ❖ Defines new fields through the use of various field-level functions
- ❖ Converts data in fields from one format to another either using internal conversion
- ❖ Maps common fields from differently formatted input files to a uniform sort record
- ❖ Joins any fields from several files into an output record, usually based on a condition
- ❖ Changes record layouts from one file type to another, including: Line Sequential, Record Sequential, Variable Sequential, Blocked, Microsoft Comma Separated Values (CSV), ACUCOBOL Vision, MF I-SAM, MFVL, Unisys VBF, VSAM (within UniKix MBM), Extended Log Format (W3C), LDIF and XML
- ❖ Maps processed records to many differently formatted output files
- ❖ Writes multiple record formats to the same file for complex report requirements
- ❖ Performs mathematical operations and functions on field data (including aggregate data) to generate new output field values

Field Reformatting/Validation

- ❖ Retrieves and re-maps values from multi-dimensional, tab-delimited lookup files on the basis of equal or conditional matches (suitable for Slowly Changing Dimensions)
- ❖ Creates and processes sub-strings of original field contents, where you can specify a positive or negative offset (from the left or right, respectively, of the source field) and a number of bytes to be contained in the sub-string
- ❖ Finds a user-specified text string in a given field, and replaces all occurrences of it with a different user-specified text string on output
- ❖ Analyzes fields to display the offset number for the specified occurrence of a string
- ❖ Manipulates and displays literal values with input data inside field statements for use in value derivations, functions, conditions, cross calculations, and reporting
- ❖ Aligns desired field contents to either the left or right of the target inrec or output field, where any leading or trailing fill characters from the source are moved to the opposite side of the string
- ❖ Supports Perl Compatible Regular Expressions (PCRE), including pattern matching
- ❖ Uses C-style "iscompare" functions to validate contents at the field level (for example, to determine if all field characters are printable), which can also be used for record-filtering via /INCLUDE and /OMIT statements
- ❖ Supports custom-defined data masks and field layout templates to structure composite elements within new (e.g., master data format) field formats for mapping and validating data

- ❖ Encrypts with user-specified functions or these built-in libraries: 3DES, AES-128 and 256 bit (including format preserving), GPG, and FIPS-compliant OpenSSL
- ❖ De- and re-identifies with built-in ASCII de-ID, binary encoding, pseudonymization, and tokenization
- ❖ Redacts with character masking or trimming, conditional field or row removal, and value substitution or randomization
- ❖ Supports custom, user-written field-level transformation libraries, and documents an examples of data cleansing routines from Melissa Data and Trillium

Record Aggregation

- ❖ Consolidates records with equal keys into unique records, while totaling, averaging, or counting values in specified fields, including derived (cross-calculated) fields
- ❖ Produces maximum, minimum, average, standard deviation, sum, and count fields
- ❖ Displays running summary value(s) up to a break (accumulating aggregates)
- ❖ Breaks on compound conditions
- ❖ Allows multiple levels of summary fields in the same report
- ❖ Re-maps summary fields into a new format, allowing relational tables
- ❖ Ranks data through a running count of descending numeric values
- ❖ Writes detail and summary records to the same output file for structured reports
- ❖ Supports multiplication
- ❖ Calculates aggregate values within sliding range windows
- ❖ Uses fuzzy lookup logic for trend reporting and predictive analytics

LICENSING INFORMATION

IRI and its expert agents around the world license **CoSort** for perpetual use on individual computer systems. Maintenance (technical support and site-specific software updates) services are provided free of charge during the first year after installation. Subsequent annual maintenance is usually offered at 15% and 20% of the base license fee, and 24/7 technical support is available as an upgrade.

CoSort license fees for Unix systems are based on specific machine make and model numbers. CoSort license fees for x86 Linux and Windows are based on RAM. In either case, additional charges for multiple CPUs and cores are assessed to reflect the performance gains from multi-threaded operations.

License fee discounts apply for multiple copies of CoSort at the same installation, and for runtime integration and redistribution (for ISVs only). IRI is generous with credit for hardware upgrades and migrations, and provides for no- or low-cost fail-over (disaster recovery) licenses.

U.S. educational and 501c(3) non-profit institutions qualify for a 10% license fee discount and government agencies will find CoSort on the GSA schedule.

A confidential license fee quotation and a free 30-day trial are available from your IRI agent, pursuant to a non-disclosure agreement.

A free, 30-day trial period is offered prior to licensing. A non-disclosure agreement must be completed, executed and returned by fax or mail to an IRI agent.

PROFESSIONAL SERVICES

An IRI professional services engagement allows you to leverage more than 100 collective years of IT and integrated data handling experience. Examples of IRI professional services engagements involve:

- Big data preparation – packaging, protecting, and provisioning structured and unstructured data sets for BI within existing DBs and file systems (and without the skills gap, high cost, and failure rate associated with Hadoop or ELT appliances)
- Data masking – profiling, de-identification, encryption, tokenization and other services to aid data loss prevention, data governance, and data privacy law compliance efforts
- Data replication and federation – acquiring, re-mapping, and creating virtualized views
- Database migration – mapping table data and relationships to new versions on platforms
- Legacy data conversion – reformatting LDIF, XML, and COBOL index files (e.g., Vision, MF-ISAM), multi-byte character sets, most mainframe data types, and endian conditions
- Master data management – value and format definition, quality validation, and security
- Program replacement – translating cryptic and inefficient SQL, 3GL, ETL and shell procedures into IRI's simple, portable, open-text 4GL scripts
- Test data management – end-to-end services from needs definition through data generation and target persistence (without using production data)

COMPANY BACKGROUND

Innovative Routines International (IRI), Inc. is an independent software vendor (ISV) specializing in data management and data protection. Better known as "The CoSort Company," IRI was founded in 1978 to pioneer commercial-grade data sorting off the mainframe.

As data volumes have grown, so has IRI. More than 35 years later, IRI and its clientele have partnered in the creation, improvement, and expansion of an integrated product line focused on:

Data Management - Data Movement, Data Integration, Data/DB Migration, and BI

Data Protection - Static Data Masking, Dynamic Data Masking, and Safe Test Data

IRI continues to grow and satisfy its worldwide customer base through its ongoing dedication to speed, ease, versatility, and value in the product line, and a culture of agility, innovation, quality and diversity in its professional services.

INNOVATIVE ROUTINES INTERNATIONAL (IRI), INC.

Suite 303, Atlantis Center
2194 Highway A1A
Melbourne, FL 32937-4932 USA
Phone +1 321-777-8889
<http://www.iri.com>



Trademarks: CoSort and FieldShield are registered trademarks of Innovative Routines International (IRI), Inc. FACT, NextForm, RowGen, SortCL and SortI are trademarks of IRI, Inc. All other brand or product names are, or may be trademarks, or registered trademarks, of their respective holders/companies.