

CobolTransformer Tools Manual

Version 3-8-4

©1995-2000 by Siber Systems

November 1, 2000

Abstract

This manual describes command-line and visual tools and converters based on **CobolTransformer**.

Short list of tools: `cbl-beau`, `cbl-grep`, `cbl-report`, `cbl2fdd`.

Short list of converters: `ibm2fsc`, `mf2fsc`, `y2k-fix`.

Copyright. © Copyright by Siber Systems 1995-2000.

No part of this software and/or documentation may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic mechanical, magnetic, optical, manual or otherwise, without the prior written permission of Siber Systems. <http://www.siber.com/>.

Inquiries.

Web Site	http://www.siber.com/sct/
Pricing, licensing, other pre-sales questions	info@siber.com
Technical support questions	support@siber.com

Trademarks. CobolTransformer, CodeTransformer and SourcePrint are registered trademarks of Siber Systems Inc.

Fujitsu, Fujitsu Software Corporation, Fujitsu COBOL and Fujitsu NetCOBOL are trademarks or registered trademarks of Fujitsu Ltd.

IBM is a registered trademark of International Business Machines Corporation.

Micro Focus is a registered trademark and Micro Focus COBOL is a trademark of Micro Focus Limited.

Microsoft, MS-DOS, Windows, Windows 95, and Windows NT are either trademarks or registered trademarks of Microsoft Corporation.

RM/COBOL is a registered trademark of Liant Software Corporation.

UNISYS is a registered trademark of UNISYS Corporation.

UNIX and X/OPEN are registered trademarks in the United States and other countries, licensed exclusively through X/Open Company Limited.

Wang is a registered trademark of Wang Laboratories Inc.

Other product names are trademarks or registered trademarks of each company.

Trademark indications are omitted for some system and product names described in this manual.

Contents

1	Running Command Line Converter Tools	3
1.1	How to Start a Tool	3
1.2	Using CL-launcher	4
1.3	Kinds of Command Line Options	4
1.4	Conversion Steps	4
1.5	SourcePrint vs PrettyPrint	5
2	Tool and Converter Options	6
2.1	Informational Options	6
2.2	Reporting Options	6
2.2.1	PrettyPrint File Generation	7
2.3	Common Options from Environment Variables	7
2.4	Cobol Parser	8
2.4.1	Cobol Dialects	8
2.4.2	Language Features	9
2.4.3	Copy Libraries	10
2.4.4	Messages and Extras	11
2.4.5	Length and Offset Computation	12
2.5	PrettyPrinter	13
2.5.1	General Options	13
2.5.2	Indentation	13
2.5.3	Printing out Comments	14
2.5.4	Line Identification	15
2.6	Examples	16
2.6.1	Cbl-Beau example	16
2.6.2	Ibm2Fsc example	16
2.7	Return Codes	16
3	Cobol Source Tools	17
3.1	Cobol Beautifier	17
3.1.1	General Options	17

3.1.2	Renumber Sections Options	18
3.1.3	Renumber Paragraph Options	18
3.1.4	Renumber Data Items (Record Description) Options	19
3.2	Cobol Grep	19
3.2.1	Options	19
3.2.2	Query Language Reference	20
3.2.3	Query Language Examples	32
3.3	Cobol Reporter	34
3.3.1	Options	34
3.4	Transformation Runner	35
3.4.1	Options	35
4	Cobol Source Converters	36
4.1	IBM Cobol to Fujitsu Cobol converter	36
4.1.1	Options	36
4.2	Micro Focus Cobol to Fujitsu Cobol converter	37
4.2.1	Options	37
4.2.2	Files Included	38
4.3	ICobol to Fujitsu Cobol converter	38
4.3.1	Options	39
4.4	Year 2000 Window Fix converter	40
4.4.1	Options	40
5	Cobol Data File Converters	41
5.1	Cobol File Descriptor Extractor	41
5.1.1	Options	41
5.1.2	FDD: File Descriptor Data	42
5.1.3	RDD: Record Descriptor Data	44
5.2	Cobol Data File Format Guess Program	48
5.2.1	Options	48
5.3	Cobol Data File To Flat File Converter	49
5.3.1	Options	49
5.4	Cobol Data File To DBase IV DBF File Converter	50
5.4.1	Options	51
5.5	Crystal Reports Cobol Data Reader DLLs	51
6	Legal	59
6.1	Paid-For and Shareware Tool License	59
6.2	Trial Tool License	62

Chapter 1

Running Command Line Converter Tools

This document describes a generic Siber Systems *Converter or Tool* that is built using CobolTransformer. The tools can be run both from command line and from the Visual CobolTransformer.

These tools have a lot in common, so we created this document that describes the common properties of these tools.

All command line tools are built using Siber Systems CobolTransformer (SCT) toolkit. If you need a toolkit to build programs like this, visit us at <http://www.siber.com/sct/>.

1.1 How to Start a Tool

Use the tool itself to get information about options that it supports. You can do it in several ways:

- `tool-name`

If you start a tool without any arguments, it will print out its version, its hard-wired options (trial or production, supported dialects, CobolTransformer link option, etc).

- `tool-name -help`

Option `-help` causes tool to print a list of command-line options that it supports. Option printouts include list of legal values and default values.

General format for tool invocation is:

```
<tool-name> -option1 -option2 -option3 ... file1 file2 file3 ...
```

This invocation would process Cobol files `file1`, `file2`, and `file3`, ..., each with options `-option1`, `-option2`, `-option3`, ...

One invocation of a tool can process multiple files. Wild-carded file names are accepted (applies only to WIN32 tools).

1.2 Using CL-launcher

CL-launcher is a GUI tool that is used to launch command line tools. It works on WIN32 systems – Windows 95, Windows 98, Windows NT.

It is used instead of MS DOS Prompt to start our converters and tools.

Its features:

- **Log Window** Converter tool output is stored in Log Window. It can be saved to text file and printed.
- **Command History** Command history is memorized in the registry.

1.3 Kinds of Command Line Options

All options start with '-' (dash). The string that does not start with dash is treated as Cobol source file name.

The followings kinds of options are accepted:

- **Boolean**

To set boolean option **xxx** to True/On specify it as **-xxx**. To set boolean option **xxx** to False/Off specify it as **-no-xxx**.

- **Number**

To set number option **xxx** to number **nnn**, specify it as **-xxx=nnn**.

- **String**

To set string option **xxx** to string **sss**, specify it as **-xxx=sss**.

- **String List**

To set string list option **xxx** to strings **s1, s2, ..., sN**, specify it as **-xxx=s1;s2;...;sN**. Semicolon “;” and comma “,” can be used as separators.

- **Number List**

To set number list option **xxx** to numbers **n1, n2, ..., nN**, specify it as **-xxx=n1;n2;...;nN**. Semicolon “;” and comma “,” can be used as separators.

1.4 Conversion Steps

Most tools perform three steps:

- **Parsing** Turn Cobol program into CobolTransformer Program Tree.
- **Conversion/Reporting** Convert the Program Tree or produce a report for the Program Tree.
- **Code generation** Generated Cobol code from the converted Program Tree. This step may be skipped for reporting-only tools such as **cbl-grep** or **cbl-report**.

1.5 SourcePrint vs PrettyPrint

The sequence of steps above is the standard approach to program conversion. The code is generated according to standard layout rules often referred to as "pretty printing" rules. Pretty-printed code removes the difficulties of interpreting untidy code, or of understanding different programmer's styles of laying out their code. Consequently, you may see this as an added benefit.

A problem of pretty-printed code is that it usually differs from the original source code. The differences are minor - words appear in different columns, and some optional words may be skipped or added. While these differences may seem insignificant, they do not allow you to use `diff`, or other automatic file comparison utility, to independently verify which lines were changed by the tool. This is because the little differences introduced in generating code from the Program Tree make every line look different to the diff utilities.

The tools therefore offer a *SourcePrint* option (set by using the `-gen-src` command line option). In this feature, the tool stores additional data about the original code in the tree nodes. This allows the tool to generate Cobol code that does not differ from the original code – down to a single byte – for the tree fragments that were not changed by conversions. Code that is generated from the transformed tree nodes is generated in the pretty-printed style.

The benefit of the SourcePrint format is that you can easily and inexpensively verify the conversions performed by the tool. When you finish a conversion, you can run a `diff` utility that compares the original sources with the converted sources. The generated difference logs reflect only the changes made by the tool, and not the changes introduced by beautification or other parsing/printing side effects.

Chapter 2

Tool and Converter Options

2.1 Informational Options

- `-help`

Displays a list of all the tool options giving the type of option (boolean, number, string, string list), the current setting, and a brief explanation. The list is displayed after all options have been accepted from the command line. Default: Off.

- `-pf`

Displays the settings of all the tool options without the type and brief explanations provided by the `-help` option. The list is displayed after all options have been accepted from the command line. Default: Off.

- `-v`

Displays the version number of the tool. Default: On, if no other options specified, otherwise Off.

2.2 Reporting Options

- `-conv-warn`

Warn about transformation problems. On by default.

- `-conv-info`

Produce informational message for every transformation performed. Off by default.

- `-find-only`

Show the places where transformations apply, but do not perform them. Please note that this might produce a list of potential transformations that would be different from actual list of transformations because certain transformations when applied make possible (or impossible) other transformations. Off by default.

- `-silent`

Do not print short summary of the conversion. Off by default.

- **-progress**

Displays how the conversion is progressing. Line numbers are displayed every 100 lines, and at the beginning of each copybook.

- **-bailout-level**

Sets the level of parser error that makes the tool bailout i.e. prevents the tool from processing the source file further. Levels are:

- * **warnings**

Warning messages give information that does not affect the conversion process.

- * **errors**

Error messages do affect the conversion but they are not severe enough to stop parsing process.

- * **severe**

Severe errors prevent the parser from continuing.

- **-stat**

Display brief source program statistics that includes number of lines in the program and the number of lines of code (LOC).

2.2.1 PrettyPrint File Generation

- **-gen-file**

Defines the filename to use for the output converted COBOL file. If no name is specified the base of the input filename is used and suffix specified in **-gen-sfx** is added to the base.

- **-gen-sfx**

Defines the extension to use for the output COBOL file. Default: depends on a particular tool.

- **-gen-main-dir**

Defines the directory to which the output COBOL file should be written. If no directory name is specified, directory containing the main input source is used.

2.3 Common Options from Environment Variables

Command line options, as the name suggests, are taken from the command line. However you may want many files to have the same set of options and it may be inconvenient to add the same options to command line for every file involved. In this case you can use environment variables **SCT_TOOL_OPTIONS** and **SCT_FPP_OPTIONS** to set the default option values that will be effective for all files.

Before parsing options from the command line, non-FPP command line tool parses options contained in the environment variable **SCT_TOOL_OPTIONS**. The individual options in **SCT_TOOL_OPTIONS** should be separated by space(s). Options enclosed in double quote characters " " can contain spaces.

Since command line options are parsed after the **SCT_TOOL_OPTIONS** options, options specified in command line may override options specified in **SCT_TOOL_OPTIONS**.

Similarly, FPP command line tools take their options from environment variable SCT_FPP_OPTIONS.

2.4 Cobol Parser

2.4.1 Cobol Dialects

- **-lang**

Specify primary Cobol dialect. Dialect can be one of the following COBOL dialects:

ansi74	1974 ANSI COBOL standard
ansi85	1985 ANSI COBOL standard
osvs	IBM OSVS COBOL
vsii	IBM VS COBOL II (revision 2, 3 and 4)
saa	IBM SAA COBOL (levels 1 and 2)
dosvs	IBM DOSVS COBOL
mf	Micro Focus
ms	Microsoft
xopen	X/OPEN COBOL
rm	RM COBOL 74
rm85	RM COBOL 85
univac	UNISYS COBOL
wang	Wang VS COBOL
icobol	ICobol
fsc	Fujitsu COBOL
net	Fujitsu NetCOBOL

Default: ANSI-85.

- **-lang2**

The secondary dialect can be specified if your program uses two different dialects, or if a dialect, such as VS COBOL II Release 2, adds on to an earlier dialect, such as OSVS. For example, to process a VS COBOL II Release 2 program use: **-lang=vsii -lang2=osvs**.

Default: none.

- **-line-format**

Source line format. Determines format of the Cobol source.

- * **fixed**

- The standard Cobol format. Columns 1 to 6 contain sequence number, column 7 contains indicator, columns 8 to 72 contain Cobol code.

- * **free**

- Free format. Column 1 contains indicator, columns 2 to 255 contain Cobol code.

- * **fsc-free**

- Fujitsu Cobol "free" format. There is no indicator area, all comments start with "***>**", Cobol code may appear in any column.

- * **var**

Variable format. Columns 1 to 6 contain sequence number, column 7 contains indicator, columns 8 to 255 contain Cobol code.

Default: determined by the dialect. For FSC Cobol default line format is variable, for ICobol it is free and for all other Cobol dialects default line format is fixed.

- **-progid-comments**

Allows comments to immediately follow the Program-ID in the PROGRAM-ID paragraph.

Default: Off.

- **-sep-space-reqd**

Generally IBM COBOL, and other dialects, require spaces after separator characters such as comma and semicolon. Some compilers have allowed the space to be omitted, for example allowing `ARRAY(I,J)`. To prevent code like this, turn this option Off: **-sep-space-reqd=no**. **-sep-space-reqd=yes** turns the checking On. **-sep-space-reqd=dial** determines whether to require spaces or not based by Cobol dialect. Currently MF, MS and ICobol dialects automatically cause **-sep-space-reqd=no**.

Default: dial.

- **-exclude-keywords**

Gives a list of COBOL words, separated by semi-colons “;” that should be excluded from the list of keywords when parsing the program to be converted. These words can then be used as user-defined words (for data item names, etc) within the program.

Default: none.

2.4.2 Language Features

- **-set-constants**

Provides the same function as the Micro Focus `CONSTANT` compiler directive. This directive allows you to use constant names in your program but set their values on the command line. The effect is as if you declared a 78-level item with that value.

Full format of the option is:

`set-constants=[const-name-1'string-value',]... [const-name-2(numeric-value),]...`

Quotes indicate the value is a string; parentheses indicate that it is a numeric value. You must use single quotes, not double quotes, and there must be no spaces between any items in the **-set-constants** option. All constants must be defined in a single **-set-constants** option.

Default: none.

- **-assign-external**

If neither `EXTERNAL` nor `DYNAMIC` are specified in the `ASSIGN TO` clause, tells the tool to assume the (Micro Focus) assignment is `EXTERNAL`. If **-no-assign-external** (default) is specified the assignment is assumed to be `DYNAMIC`.

Default: Off.

- **-sql**

Parse SQL in EXEC SQL statements. If this option is Off, EXEC SQL statements are passed thru as a sequence of tokens. If this option is On, EXEC SQL statements are parsed into a tree, so that they can be analyzed and transformed.

Default: On.

- **-cics**

On: Parse CICS statements embedded into EXEC CICS ... END-EXEC. Off: let CICS go through as a sequence of tokens.

Default: Off.

- **-cics-eib**

Add CICS EIB data block at the beginning of LINKAGE SECTION.

Default: Off.

2.4.3 Copy Libraries

- **-copylib-dir**

Specify list of directories searched for COPY library files.

Directory names can be absolute or relative to the working directory.

Directory names are separated by semicolons ";" or comma ",".

Directory names that contain spaces must be enclosed in quotes " ".

Both "/" (slash) and "\" (backward slash) can be used as separators of individual directory names in full directory names.

If library is specified explicitly:

`COPY file OF library.`

then `library` is used as the name of environment variable. This variable must contain the directory name in which this library is located.

Default: only the current working directory (".") is searched.

- **-copylib-sfx**

Defines the extension (suffix) to append to COPY library file names when the extension is not specified in the program source (as in "COPY file."). When file name suffix is explicitly specified, as in "COPY "file.ext"." then the default suffix is not added. Note that the period (".") needs to be specified as part of the extension string.

If `-copynames-case=lower` and COPY file "FiLe.SFX" is not found, try lowercase version "file.sfx" of the file name.

If `-copynames-case=upper` and COPY file "FiLe.sfx" is not found, try uppercase version "FILE.SFX" of the file name.

If `-copynames-case=exact` then COPY file name should match exactly the actual file name.

Please note that on WIN32 platforms file name capitalization is ignored by the system, so `-copynames-case=exact` will work just fine. This means that using this options on WIN32 will not change anything.

Debugging this: if you want to monitor the tool trying to locate copybooks and other file-related activities, set environment variable `SCT_DEBUG` to 1.

Default: `.cpy`

- **-copynames-case**

If a `COPY` file cannot be located, gives the character case the tool should use for the file name in further open attempts: all **upper** case, all **lower** case, or **exactly** as coded.

Default: `lower`.

- **-old-copy**

Indicates that the program to be converted uses ANSI 68 format `COPY` statements. In this format, the name before the word `COPY` was used to replace the name at the equivalent level in the text file.

Default: `Off`.

- **-irrev-inline**

When the **-irrev-inline** option is `Off` (`-no-irrev-inline`) the lexical analyzer inserts `COPY` and `ENDCOPY` tokens in the token stream indicating the beginning and end of copybook code. With these tokens, the parser can place `COPY` statements on a Program Tree and therefore it can output separate copybook files or place enter and exit copybook comments in the COBOL file (controlled by the `-gen-copy-dir` and `-gen-enter-exit-copy-comments` options).

However, certain Cobol structures involving copybooks cannot be parsed with the `COPY` and `ENDCOPY` tokens in the token stream, because in general case, `COPY` and `ENDCOPY` tokens are not part of the Cobol grammar. To parse these structures, or if you always want the copybook code expanded inline, set the `-irrev-inline` option.

Default: `Off`.

- **-inline-copy**

If `On`, parse copybooks. If `Off`, ignore copybooks as if they were not found.

Default: `On`.

2.4.4 Messages and Extras

- **-warnings**

Display warning messages.

Default: `On`.

- **-multi-undefd-errs**

Indicates that an error message should be output for every use of an undefined name. When **-no-multi-undefd-errs** (default) is set, only one error message is issued on the first use of the undefined name.

Default: `Off`.

- **-resolve-use-def**

Resolve use-definition links for named objects.

Default: On.

- **-same-para-data-name**

-same-para-data-name=yes allows one name to be used both as paragraph-name and data-name. -same-para-data-name=no disallows this behaviour as required by standards. -same-para-data-name=dial specifies that Yes/No is determined by the dialect. Currently for MF and ICobol this property is set to Yes, and for all other dialects it is set to No.

Default: dial.

2.4.5 Length and Offset Computation

- **-leng-offs**

Compute data item lengths and offsets.

Default: On.

- **-lo-stor-mode**

Type of alignment (storage mode) used to compute length and offset of data items. If it is **byte**, the binary items length can be any number of bytes from 1 to 8 and they can be aligned on the byte boundary. If it is **word**, the binary items length can be 2, 4, or 8 bytes and they are aligned on 2, 4, or 8-byte boundary. If storage mode is **dial**, then the actual storage mode is determined by the dialect.

Default: **dial**.

- **-num-sign-trail-sep**

Flag: Is Numeric Sign a Trailing Separate Character. -num-sign-trail-sep=no means No. -num-sign-trail-sep=yes means Yes. -num-sign-trail-sep=dial leaves it to the dialect to determine whether the answer is Yes or No. Currently only RM Cobol dialects have this option set to Yes.

Default: dial.

- **-lo-pointer-size**

Size of the POINTER data item.

Default: 4.

- **-lo-proc-pointer-size**

Size of the PROCEDURE-POINTER data item.

Default: 4.

- **-lo-index-size**

Size of the INDEX data item.

Default: 4.

- **-lo-unfold-flex-arrays**

If this option is On, then the size of table with **DEPENDING ON** is computed as difference between upper and lower bound on indices. If this option is Off, then size table with **DEPENDING ON** is undefined.

Default: On.

2.5 PrettyPrinter

2.5.1 General Options

- **-gen-src**

Indicates whether the output source should be in SourcePrint format (**-gen-src**), which preserves the format of all but the converted lines, or in PrettyPrint format (**-no-gen-src**) in which source is fully reformatted (beautified) in a regular manner.

Default: Off.

- **-gen-copy-dir**

Indicates that copy files should be written to separate files, rather than being expanded (inlined) in the main COBOL file, and defines the directory to which the output copy files should be written.

If set to a null string (**-gen-copy-dir=""**, the default), copy files are expanded inline in the output file. Only valid if the **-no-irrev-inline** option is set.

Copybook name that is stored in the Program tree is appended to the directory name specified in **-gen-copy-dir** option to form full name of output copybook file. However, if copybook name stored on the tree is absolute name (this happens if you specify absolute names in **-copylib-dir** option), then only base file name derived from this absolute copybook name is appended to output copybook directory name.

If you use **".."** (parent directory) in directory name specified in **-copylib-dir**, then substrings **xxx/..** where **xxx** is arbitrary directory will be removed from the resulting copybook name, because WIN32 systems do not accept **".."** in file name.

2.5.2 Indentation

- **-gen-indent-step**

When the tool outputs pretty-printed code it uses indentation to help the interpretation of statements that are spread over multiple lines or have nested levels of logic (such as nested IF statements). The **-gen-indent-step** option specifies the number of character positions by which each successive indent is displaced from the previous indent.

Default: 2.

- **-gen-max-indent**

Defines the maximum column at which indented text will be started. Any text whose indent would start at a column greater than this column is output with the indent starting at this column.

Default: 40.

- **-gen-tabs**

When the Converter outputs pretty-printed code certain items, such as PICTURE clauses, are aligned to tab columns. The **-gen-tabs** option specifies the tab column positions.

Currently tabs are used for data items and report section items:

Data items	Tab
PICTURE	\#4
REDEFINES	\#4
USAGE	\#6
OCCURS	\#4
VALUE	\#6

Report section items	Tab
PICTURE	\#3
SOURCE	\#5
VALUE	\#5

Default: 12,24,32,42,44,54.

- **-gen-line-format**

Generated line format. Can be: **fixed** for fixed format, **free** for free format, **fsc-free** for Fujitsu free format, or **var** for variable format. See **-line-format** option description for more information on line formats.

Default: fixed.

- **-gen-observe-ab-rules**

If set to False, then disregard the Area A, Area B rules when generating converted code. Ignored if **gen-line-form** is **free**.

Default: Off.

2.5.3 Printing out Comments

- **-gen-print-comments**

Comments from the input source are preserved in the output source.

Default: On.

- **-gen-enter-exit-copy-comments**

Indicates that the tool should place == Enter/Exit filename == comments before and after copybook text that is expanded in the main output file. Only valid if the **-no-irrev-inline** option is set.

Default: On.

- **-gen-add-line-dirs**

Generate FSC preprocessor LINE/FILE directives, so that FSC compiler and debugger report errors and debug in terms of the source Cobol file, not in terms of the converted file.

Default: On.

2.5.4 Line Identification

- **-gen-line-id-comments**

Indicates that the tool should transfer characters held in columns 1-6 and 73-80 from the input source to the output source. Effective only when **-gen-src** is On.

Default: On.

- **-gen-73to80-fmt**

Specify a string to be placed in columns 73 to 80. You can include the line number in the string by including one of the following "%" strings (a la C printf):

- * **%d**

output the line number using the minimum number of digits (i.e. one digit for the numbers 1-9, two digits for the numbers 10 to 99, etc.),

- * **%nd**

use **n** digits for the line number (right aligned, space filled),

- * **%-nd**

left align the line number in **n** digits,

- * **%0nd**

zero fill a right aligned, **n** digit line number.

You can also substitute the letters "o" or "x" for the letter "d" in the above strings to output the numbers in octal or hexadecimal.

Some sample formats:

- * **-gen-73to80-fmt="ABC%5d"**

In the first three lines the following would be inserted to columns 73 to 80:

```
ABC    1
ABC    2
ABC    3
```

- * **-gen-73to80-fmt="%06dYZ"**

In the first three lines the following would be inserted to columns 73 to 80:

```
000001YZ
000002YZ
000003YZ
```

Default: no format.

- **-gen-73to80-start**

Number from which to start the numbers placed in columns 73 to 80.

Default: 1.

- **-gen-73to80-step**

Amount by which the numbers placed in columns 73 to 80 are incremented from one line to the next.

Default: 1.

2.6 Examples

2.6.1 Cbl-Beau example

To beautify an OSVS Cobol program that is named prog11.cbl type:

```
cbl-beau -lang=osvs prog11.cbl
```

File prog11.scc is created. It contains the beautified program.

2.6.2 Ibm2Fsc example

To convert Cobol program z80.cbl written in VS Cobol II to Fujitsu Cobol (the source program has comments in PROGRAM-ID entry, copy libraries are in directories "copylib" and "include", copylib files suffix set to ".cfx"), type the following:

```
ibm2fsc z80.cbl -lang=vsii -progid-comment -copylib-dir=copylib;include -copylib-sfx=.cfx
```

2.7 Return Codes

Tools return integer *Return Codes* that can be checked in MS DOS batch (errorlevel) or UNIX shell files.

RetCode	Meaning
0	Conversion completed with no errors equal to or greater than the bailout level.
1	Bailout because of a warning error (only occurs if -bailout-level = warnings)
2	Bailout because of a non-terminal parser error (only occurs if -bailout-level = errors)
3	Bailout because of a severe parser error (occurs for any -bailout-level setting)
5	No action taken (e.g. only the option list was displayed)
10	Cannot create an output file
11	Cannot open an input file
12	Cannot remove a file (client-server mode only)
13	Cannot start a file (client-server mode only)
20	Option not available in installed package
21	Incorrect command line option specified
22	Incorrect dialect specified in the -lang or -lang2 option
23	Stack or array overflow
24	Error while trying to check license
Other	
Values	Internal error - contact technical support.

Chapter 3

Cobol Source Tools

3.1 Cobol Beautifier

`cb1-beau` beautifies Cobol code and performs other important code maintenance tasks.

Beautification is controlled by the general PrettyPrinter tool options. Optional code maintenance tasks are invoked by their respective options.

By default the converted file has extension `.scc`.

3.1.1 General Options

- `-add-value-clause`

Add `VALUE` clause to `WORKING-STORAGE` section data items that do not have `VALUE` clause but have `PIC` clause.

Having `VALUE` clause for every data item in `WORKING-STORAGE` section is considered important, it helps to increase program portability and may prevent some unpleasant core-dumps/GPFs.

Default: Off.

- `-norm-dd-levels`

Normalize data item level numbers so that all data items at one logical level of record hierarchy have the same level number.

Some Cobol dialects (IBM OSVS, MF) allow level numbers to be different for items belonging to the same hierarchy level. Transformation invoked by this option fixes this non-standard behaviour.

Default: Off.

- `-add-end-stmts`

Add `END-IF`, `END-SEARCH`, `END-EVALUATE`, `END-PERFORM` closing statements.

Most Cobol compilers can guess where the composite statement should end, so you are not required to have the closing `END-*` statements.

However it is considered a good programming style to have all composite statements terminating `END-*` statements coded explicitly. Transformation invoked by this option automatically adds `END-*` closing statements.

Default: Off.

3.1.2 Renumber Sections Options

- **-section-name-fmt**

Section name format, looks like “t1%dt2%st3”.

If this option is specified, then all definitions and uses of section names are renumbered. The option string specifies C-printf-like format of the new name assigned to each section.

First data item in this format is “%d”. It is replaced with section Counter. The counter is increased by the counter step for every new section that passes through the tool.

Second data item in this format is “%s”. It is replaced with the old (before renaming) section name.

t1, t2 and t3 is any text not containing percent sign. Modified forms of “%d” such as “%04d” be used.

Example1: `-section-name-fmt="%04d-%s"`. Add 4-digit (leading zeros) numeric prefix to the section name.

Example2: `-section-name-fmt="S%d-%s"`. Add character 'S' followed by section counter to the section name.

Default: no section renumbering.

- **-section-name-start**

Start value for number in section name.

Default: 1.

- **-section-name-step**

Step for number in section name.

Default: 1.

3.1.3 Renumber Paragraph Options

- **-para-name-fmt**

Paragraph name format.

Formatting rules are the same as for section names.

Default: no renumbering.

- **-para-name-start**

Start value for number in paragraph name.

Default: 1.

- **-para-name-step**

Step for number in paragraph name.

Default: 1.

3.1.4 Renumber Data Items (Record Description) Options

- **-data-name-fmt**
Data name format.
Formatting rules are the same as for section names.
Default: no renumbering.
- **-data-name-start**
Start value for number in data name.
Default: 1.
- **-data-name-step**
Step for number in data name.
Default: 1.

3.2 Cobol Grep

cbl-grep takes all the files specified by the user, parses them into Cobol Program Tree, searches the Program Tree for nodes that satisfy user-specified query and finally print out the source lines that correspond to the found tree nodes.

Query is written in CobolTransformer Query Language (QL). This language can also be called "SQL for Program Trees". Reference manual for Query Language (QL) is available in the "Query Language" Chapter.

Typical uses:

```
cbl-grep -q=query file1.cbl file2.cbl file3.cbl ...  
cbl-grep -qf=query-file:query file1.cbl file2.cbl file3.cbl ...
```

3.2.1 Options

- **-q**
Specify the query directly.
Do not forget to put query text in quotes if it contains spaces or special characters. Otherwise you shell will break it into several options and cbl-grep will not be able to parse it.
Default: no direct query specified.
- **-qf**
Read the query from the specified file.
If string specified in **-qf** option does not have colon ':', then this string is file name and this file contains exactly one query.
If **-qf** option string contains colon ':' such as in **aaa:bbb** then text before the colon (**aaa**) is treated as the file name and the text after the colon (**bbb**) is treated as the name of the query in the file. In this case every query in the file **aaa** must start with **QUERY query-name** line.
Default: do not read query from file.

- -print-query

Print query exactly as the query parser sees it. Use this option to verify that `cbl-grep` understands your query.

Default: Off.

- -print-err-pos

If query is incorrect, indicate position of error in the query.

Default: On.

- -print-tree

Print subtree that starts from the node that matches query.

Default: Off.

- -print-src

Print source line that corresponds to the node that matches query. If Off, then print only coordinates of the found nodes in the source.

Default: On.

3.2.2 Query Language Reference

Introduction

When reengineering program in computer language such as Cobol, we often need to locate certain units of program that satisfy criteria that we determine.

Example 1: For instance, we may be interested in seeing all CALL and CHAIN statements for a given program, because these statements show what external programs are called from this program.

In QL this query is written as: `E.OP==STMT_CALL OR E.OP==STMT_CHAIN` or even (simplified version): `STMT_CALL OR STMT_CHAIN`

Example 2: Or we may need to find all data item declarations that belong to FILE SECTION or LINKAGE SECTION and that have VALUE clause. It would be useful if we are migrating from the Cobol compiler that allows VALUE clauses for such data items but treats them as documentary to the Cobol compiler that does not allow VALUE clauses in such data items at all.

In QL this would be:

```
E.OP=DECL_DD_ENTRY AND
(E.ISDESCOF(SECTION_WORKING_STORAGE) OR
E.ISDESCOF(SECTION_LINKAGE)) AND
E.ARG(DD_VALUE) != NULL
```

Query language described in this document is used
to formulate such search criteria.
Query is a predicate calculated on Program Tree node:

logical_value = Query (e)

So Query receives Cobol Program Tree node e as an argument and
it returns True if node e satisfies condition encoded in the Query.
The Query returns False if condition is not satisfied by e.

BNF used for QL syntax description

We use Backus Naur Form (BNF) to describe syntax
of Query Language (QL).

The Query consists of tokens.

Non-terminal tokens appear as all lower-case words in BNF rules.

Terminal tokens appear as all upper-case words in BNF rules and
they are enclosed in single quotes like this one: 'NOT'.

Words in query that match terminals in case-insensitive way,
that is, instead of 'NOT' one can use 'not', 'NoT', 'NOt' and so on.

Query Library File

Queries contain a number of characters that have
special meaning in OS shells (space , = ! ~ etc).
Therefore it makes sense to have queries loaded from files
in addition to specifying them in command line.

Since most queries are relatively short,
we can have more than one query into a file.
It also improves query management.

So, several queries may be put together in a library file.
The queries are not compiled in the library,
they appear in the source form.

In the library of queries every query must be named.

However, if file has only one query, query name may be skipped.

Every query in a query library file has header line that has format:

QUERY query-name

Here the word QUERY must be the first word on the line.

The body of the query starts on the line that immediately follows the query header line and ends on the line that immediately precedes the next query header line or end of file.

The individual query in a library is referenced by query-locator described below.

Query Lexical Rules

Query lines that starts with characters '#' or '*' are comment lines that are ignored.

If two slashes '/' are found in the line, these slashes and text after them (till the end of line) is a comment which is ignored.

White space characters ' ', '\n', '\r', '\t' are used as delimiters for tokens. That is, token is a sequence of non-whitespace characters delimited by whitespace characters.

However, when it is possible to separate tokens, whitespace between them is not required.

Any combination of name or number or string token and any of miscellaneous token is separatable.

Any other combination (name and name, name and number, number and number, etc) must be separated by whitespace.

The following tokens are used:

name: sequence of characters 'A'-'Z', 'a'-'z', '0'-'9', '-' where first character is letter.

int-const: sequence of characters '0'-'9'.
represents integer constant.

str-const: sequence of any characters enclosed in single quotes ' or double quotes "
with following translations inside:


```

\" -> "
\' -> '
'' -> ' if string starts with '
"" -> " if string starts with "
represents string constant

```

Miscellaneous tokens:

```

( ) = . !
== > >= < <= ~
+ - *
represent various operations

```

Names of Things

Meaning of the name-like terminals:

**** query-locator**

The individual query in a library is referenced by query-locator string that has the following format:
file-name:query-name

Here file-name is a regular UNIX/MSWIN file name that can contain characters 'a'-'z', 'A'-'Z', '0'-'9', '.', '\', '/', '-'.

It also may contain character ':' in the 2nd or 3rd column of file-name if file-name starts with one of the following

[a-zA-Z]\:

[a-zA-Z]:

that is, when file-name starts with WIN32 drive designator.

It means that we disallow file-name to be one-character name with no extension.

Examples of correct query-locators:

cobc\qry.lib:qry-member

d:\cobolc\sct\QUERY-EXAMPLE.qry:my_query

/volume/dir/qrys/my-qry

**** oper**

Oper is a name-like token that represents symbolic Operation Name. Every node in Program Tree has an operation that identifies the type of node. List of all Program Tree operations is available from

the CobolTransformer Program Tree Manual.

Operation name is case-insensitive.

**** clause-index**

Clause-index is a name-like or integer-like token that represents index of child in the positional Program Tree node.

Children of every node in Program Tree are numbered from 1 to N, where N is the number of children in the node.

For positional nodes every child represents a clause, so it is convenient to have symbolic names for this clauses. Symbolic clause-index is such a name.

The list of symbolic clause names is available from the CobolTransformer Program Tree Manual.

Integer child index can be used instead of symbolic child index for positional nodes with unnamed children or for children of list nodes which are 'naturally' unnamed.

Clause name is case-insensitive.

**** node-var**

Node-var is a name-like token that represents a name of node variable. Each node variable must be explicitly declared before its use.

**** value-var**

Value-var is a name-like token that represents a name of value variable. Each value variable must be explicitly declared as either string variable or integer variable.

Therefore, there are three distinct types of variables in QL:

- Node Variable: points to Program Tree nodes,
- Integer Value Variable: holds integer value,
- String Value Variable: holds string value.

Individual Query

```

query -- query
node  -- expression that points to a Program Tree node
value -- integer or string value (constant or extracted from a node)

query: 'TRUE'
      | 'FALSE'

      // node reference points to non-existing node (node reference is NULL) or
      // pointed to node is NULL node.
      | node '==' NULL

      // node reference points to existing node (node reference is not NULL) and
      // pointed to node is not NULL node.
      | node '!=' NULL
      // equivalent to: node != NULL
      | node

      // equivalent to: node.OP == oper
      | node '==' oper

      // equivalent to: node.OP != oper
      | node '!=' oper

      // Return True if 1st and 2nd node expressions point to the same node
      | node '==' node

      // Return True if 1st and 2nd node expressions point to different nodes
      | node '!=' node

      // return True if node is a descendant of node with operation oper
      | node '.' 'ISDESCOF' '(' oper ')'

      // equivalent to: E.ISDESCOF(oper)
      | 'ISDESCOF' '(' oper ')'

      // Return True if this node originates from Cobol source.
      // Return False if this node was generated (does not come from the source).
      | node '.' FROMSRC

      // Return True if 1st operation is equal to the 2nd operation
      | oper '==' oper

      // Return True if 1st operation is not equal to the 2nd operation
      | oper '!=' oper

      // equivalent to: E.OP == oper

```

```

| oper

    // is 1st value equal to 2nd value, case-sensitive strings and integers
| value '==' value

    // is 1st value equal to 2nd value, case-insensitive strings
| value '==~' value

    // equivalent to: value ==~ value
| value '~' value

    // is 1st value not equal to 2nd value, case-sensitive strings and integers
| value '!=' value

    // is 1st value not equal to 2nd value, case-insensitive strings
| value '!=~' value

    // is 1st value >= 2nd value, case-sensitive strings and integers
| value '>=' value

    // is 1st value >= 2nd value, case-insensitive strings
| value '>=~' value

    // is 1st value > 2nd value, case-sensitive strings and integers
| value '>' value

    // is 1st value > 2nd value, case-insensitive strings
| value '>~' value

    // is 1st value <= than 2nd value, case-sensitive strings and integers
| value '<=' value

    // is 1st value <= than 2nd value, case-insensitive strings
| value '<=~' value

    // is 1st value < 2nd value, case-sensitive strings and integers
| value '<' value

    // is 1st value < 2nd value, case-insensitive strings
| value '<~' value

    // does 1st value match pattern from the 2nd value
| value 'LIKE' value

    // Return True if query is False.
    // Otherwise return False.
| 'NOT' query

```

```

| '!'    query

    // If 1st query is False, return False.
    // Otherwise evaluate 2nd query and return its value.
| query 'AND' query
| query '&&' query

    // If 1st query is True, return True.
    // Otherwise evaluate 2nd query and return its value.
| query 'OR'  query
| query '||'  query

    // return value of query
| '(' query ')';

    // call query specified by QUERY_LOCATOR with arg node
| query-locator '(' node ')';

```

Query takes in the main variable which is reference to Program Tree node called E. Based on node E, Query must compute logical True/False value and return it.

Query may refer to variables other than E. These variables:

- May be declared in the query.
Then they are called Query Variables.
- May be declared outside of query.
Then they are called Free Variables.

Main variable, Query Variables and Free Variables all point to Program Tree nodes. References to them appear as non-terminal 'node' in Query BNF.

When Query is evaluated, it receives values of all variables, including main variable and free variables.

Query variables are assigned value inside the query.

Comparison operators and assignment operators:

We follow C, C++ and Java tradition and use

'=' as assignment operator and
'==' as equal-to comparison operator.

LIKE operator matches its 1st argument against the pattern taken from the 2nd arg. The RX package is used for pattern matching, and format of pattern string is described there.

Empty string query is equivalent to 'TRUE' query, that is,
to query that returns True for any Program Node presented to it.

```
node: // Return value of the main node variable
      'E'

      // Return the Root variable.
      // This is a root of the Program Tree in which search occurs.
| 'ROOT'

      // Return value of the previously declared node variable
| node-var

      // assign node's value to the previously declared variable var-node
| node-var '=' node

      // declare node variable var-node and assign node's value to it
| 'NODE' node-var '=' node

      // return parent of the node
| node '.' 'PARENT'

      // equivalent to: E.PARENT
| 'PARENT'

      // return child/argument number clause-index of the node
| node '.' 'ARG' '(' clause-index ')'
| node '.' 'CHILD' '(' clause-index ')'

      // equivalent to: node.ARG(CLAUSE_INDEX_CONSTANT)
| node '.' CLAUSE_INDEX_CONSTANT

      // equivalent to: E.ARG(clause-index)
| 'ARG' '(' clause-index ')'
| 'CHILD' '(' clause-index ')'

      // Find 1st occurrence of operation oper among arguments of node.
      // If not found, return NULL.
| node '.' 'FIND' '(' oper ')'

      // equivalent to: E.FIND(oper)
| 'FIND' '(' oper ')'

      // Return Linked-to node for node thatfor which HasLink() is True.
      // Returns UNDEFINED for all other nodes.
```

```

    // Returns NULL if object has no definition.
| node '.' 'LINK'

    // Enclosing statement for the node
| node '.' ENCL-STMT

    // Enclosing section for the node
| node '.' ENCL-SECTION
;

```

Non-terminal node represents reference (pointer) to Program Tree node. These pointers may have NULL values for different reasons. For instance, when data item declaration does not have a certain clause, the child that is responsible for this clause is NULL.

If attempt is made to dereference NULL node pointer (for instance, in node.PARENT operation), the dereferencing does not occur. If node pointer is expected then NULL pointer is returned.

E represents main variable of a query.

Query Variables that are declared in the query are visible only in this query. If variable needs to be visible outside of the query (that is, in Scripting Language), it must be declared in SL.

```

value: // Integer constant
| int-const

    // String constant
| str-const

    // Image of the value stored in the node (string)
| node '.' 'IMAGE'

    // Normalized Image of the value stored in the node (string)
| node '.' 'VALUE'

    // Integer value of the node (valid only for INT_NUM_CONST node)
| node '.' 'INTVALUE'

    // Number of arguments/children of the node
| node '.' 'ARGNO'

    // Return value of integer/string variable var-value

```

```

| var-value

    // Assign value to previously declared integer/string variable var-value
| var-value '=' value

    // Declare integer variable var-value and assign value to it
| INT var-value '=' value

    // Declare string variable var-value and assign value to it
| STR var-value '=' value

    // Return length of string value
| value '.' 'LENGTH'

    // Return substring of string value that starts
    // at integer position value-1 (first character has position 0) and
    // has length of integer value-2
| value '.' 'SUBSTR' '(' value-1 ',' value-2 ')'

    // Return position of 1st occurrence of string value-1 in value.
    // First position in the string has number 0.
| value '.' 'pos' '(' value-1 ')'

    // Return position of i-th occurrence of string value-1 in value.
    // First position in the string has number 0.
| value '.' 'pos' '(' value-1 ',' i ')'

    // add two integers or concatenate two strings
| value '+' value

    // subtract two integers
| value '-' value

    // multiply two integers
| value '*' value

    // divide two integers
| value DIV value

    // compute remainder of division for two integers
| value MOD value

    // lower-case the string
| value '.' LOWER

    // upper-case the string
| value '.' UPPER

```


;

If attempt is made to dereference NULL node pointer
(for instance, in node.OP operation), the dereferencing does not occur.
If operation or value are expected to be returned as a result
of dereferencing (as in node.OP), special UNDEFINED value is returned.

The UNDEFINED value behaves by the following rules:

```
value == UNDEFINED -> FALSE
value != UNDEFINED -> TRUE
value > UNDEFINED -> UNDEFINED
value >= UNDEFINED -> UNDEFINED
value < UNDEFINED -> UNDEFINED
value <= UNDEFINED -> UNDEFINED
value + UNDEFINED -> UNDEFINED
value - UNDEFINED -> UNDEFINED
value * UNDEFINED -> UNDEFINED
value DIV UNDEFINED -> UNDEFINED
value MOD UNDEFINED -> UNDEFINED
value && UNDEFINED -> UNDEFINED
value || UNDEFINED -> value
! UNDEFINED -> UNDEFINED
```

Operation priorities from highest to lowest:

```
. 'FUNC'
* / DIV MOD
+ -
comparison opers
=
```

operations are computed left to right, that is
 $a * b * c = (a * b) * c$

Developer API

If you purchased a complete CobolTransformer distribution,
see file sct/query.h for detailed instructions
on how to use Developer API for Query Language.

Not implemented

As of Version 3.2.1 of QL, the following constructs are not implemented:

- * Query: QUERY_LOCATOR '(' node ')'
- * Query Variables
- * Free Variables

3.2.3 Query Language Examples

```
#
# This is Query Library file that contains examples of useful queries.
# These queries are used in conversion tools based on CobolTransformer.
#

#
# Find TRANSFORM statements
#
Query TransformStatement
  stmt_transform

#
# Find declaration of data item that has VALUE clause
#
Query DDhasValueClause
  DECL_DD_ENTRY and
  arg(DD_VALUE) != NULL

#
# Find data item declaration with no PICTURE clause
#
Query DDnoPICclause1
  e.op==DECL_DD_ENTRY and
  e.arg(DD_PICTURE) == NULL

Query DDnoPICclause2
  DECL_DD_ENTRY and
  Child(DD_PICTURE) == NULL

#
# Find PLUS/MINUS operations used in table references in the following way:
#   TABLE (index - +1) or TABLE (index - -1) or
#   TABLE (index + +1) or TABLE (index + -1)
#
Query TableRefIndexPlusMinusOrMinusPlus
```

```

(OP_ADD_IX or OP_SUB_IX) and
parent==SUBSCRIPT_LIST and
parent.parent==TABLE_ELT and
e.ArgNo==2

#
# Find File Declaration (FD) with REPORT clause and
# no clause that specifies record size (RECORD CONTAINS or RECORD IS VARYING)
#
Query FDwithReportClauseAndWithNoRecordContains
DECL_FD and
e.Arg(FDH_ENTRY).Arg(FD_REPORT) and
e.FDH_ENTRY.FD_RECORD_SIZE == null

#
# Find data item declarations that have VALUE clause and
# belong to WORKING-STORAGE or LINKAGE sections.
#
Query DDhasValueInWorkingOrLinkageSection
E.OP==DECL_DD_ENTRY AND
(E.ISDESCOF(SECTION_WORKING_STORAGE) OR
E.ISDESCOF(SECTION_LINKAGE)) AND
E.ARG(DD_VALUE) != NULL

#
# Find all code which affects data item "WSS",
# either directly or thru REDEFINES.
# See the power of QL!
#
Query WhatAffectsDataItem
( ( e==NAMED_OBJ_DEF && e.value~"WSS" && // Definition
  // Save section flag: only one of assignment
  // will be executed.
  ((e.isdescof(section_working_storage) && int sect=1) ||
   (e.isdescof(section_linkage) && int sect=2)
  )
  &&
  // save beginning and end of definition
  int base=e.parent.dd_offset_global.intvalue &&
  int len=base+e.parent.arg(dd_length).intvalue
)
||
( e==NAMED_OBJ_USE && // Usage
  // In the same section?
  ((e.def.isdescof(section_working_storage) && sect==1) ||
   (e.def.isdescof(section_linkage) && sect==2)
  ) &&

```

```

// Overloaded?
( int head=e.link.parent.dd_offset_global.intvalue and
  int tail=head+e.link.parent.dd_length.intvalue &&
  ( (head>=base && head<=len) || (tail>=base && tail<=len) )
) &&
// Does this USE belong to receiving part of
// MOVE, ADD, SUB, MUL, DIV?
( e.isdescof(ex_list) || e.isdescof(ex_ident_list) ||
  e.isdescof(ex_arith_div_remainder)
)
)
)

```

3.3 Cobol Reporter

`cbl-report` produces reports that summarize important properties of a Cobol program.

This tool only analyzes a Cobol program, it does not convert it.

3.3.1 Options

- `-long-stat`

Produce long program statistics that includes:

- * Number of lines in the program, Lines of code (LOC), number of tokens. All these counts include inlined copybooks.
- * Program-IDs and ENTRYs defined in this file.
- * Number of sections, paragraphs, sentences and statements in the program.
- * Number of `File IO` and `Sort-Merge` subsystem statements used.
- * Number of `SQL` and `CICS` subsystem statements used.
- * Number of `Report Writer` subsystem statements used.
- * Number of `Accept/Display ANSI` and `Accept/Display Screen (ADIS)` subsystem statements used.
- * List of all programs called from this file, with count of calls for each called program.
- * List of all `EXEC` subsystems called from this file, with count of calls for each `EXEC`.
- * List of all copybooks used in this file, with count of use for each copybook. Number of `COPY REPLACING` uses.

Default: On.

- `-mem-offsets`

Print data item length and offset information for all data items in `WORKING-STORAGE`, `FILE`, and `LINKAGE` sections that includes:

- * Level of the data item.
- * Name of the data item.

- * PICTure clause for the data item.
- * Category of the data item: A for alphabetic, N for numeric, AN for alpha-numeric, AE for alpha-numeric edited, NE for numeric edited, EF for external float, B for bit, JP for national, JE for national edited.
- * Effective Usage for the data item.
- * '*' if the data item is Synchronized.
- * Offset of the data item from the 01-level data item start.
- * Length of the data item in bytes.
- * Offset of the data item from the enclosing section start.

For 78-level constants their computed value is displayed.

Default: On.

3.4 Transformation Runner

`runtrans` loads transformation from file and runs it on specified files.

3.4.1 Options

- -t

Name of the *.SCT file that contains transformation to be run.

Default: must always be present.

Chapter 4

Cobol Source Converters

4.1 IBM Cobol to Fujitsu Cobol converter

`ibm2fsc` converts IBM Cobol dialects OSVS, VS/II, SAA, DOSVS to Fujitsu Cobol.

By default the converted file has extension ".fsc".

The complete list of conversions performed can be found in the file `SPEC.txt`.

4.1.1 Options

- `-add-value-clause`
Add VALUE clauses to WORKING-STORAGE SECTION data items that do not have VALUE clause and have PIC clause.
Default: Off.
- `-comment-execs`
Comment out EXEC ... END-EXEC statements.
Default: Off.
- `-expand-abbrev-conds`
Expand abbreviated combined conditions.
Default: Off.
- `-set-max-report-len`
User defined maximum report-record length.
Default: -1/none.
- `-fsc-options`
FSC compiler options.
Default: "SRF(FIX),BINARY(BYTE)".
- `-ibm-printer-files`
List option: `-IBM-printer-files=[printer-file-name,...]`

This will indicate the names of the files that should be handled as IBM printer files using the filenames defined in the SELECT statement.

Default: empty list.

- **-ibm-printer-char**

Sets the action to be taken for IBM printer files

Actions are:

- * **adv**
- * **noadv**
- * **none**

4.2 Micro Focus Cobol to Fujitsu Cobol converter

mf2fsc converts Micro Focus Cobol, Microsoft Cobol and Ryan McFarland Cobol to Fujitsu Cobol.

By default the converted file has extension ".fsc".

The complete list of conversions performed can be found in the file **SPEC.txt**.

4.2.1 Options

- **-add-value-clause**

Add VALUE clauses to WORKING_STORAGE SECTION data items that do not have VALUE clause and have PIC clause.

Default: Off.

- **-convert-78s**

If On, convert 78-level constants using one of the methods specified below. If Off, do not convert 78-level constants.

Default: On.

- **-propagate-78s**

If On, Propagate 78-level constants. If Off, generate REPLACE statement that defines 78-level constants.

Default: Off.

- **-iostatus-conv-only**

Perform IO status conversion only, turn off all other transformations.

Default: Off.

- **-iostatus-conv**

Perform IO status conversion, in addition to other transformations.

Default: Off.

- **-add-end-stmts**

Add END-IF, END-SEARCH, END-EVALUATE, END-PERFORM closing statements.

Most Cobol compilers can guess where the composite statement should end, so you are not required to have the closing END-* statements.

However it is considered a good programming style to have all composite statements terminating END-* statements coded explicitly. Transformation invoked by this option automatically adds END-* closing statements.

Default: Off.

- **-fsc-options**

FSC compiler options.

Default: "SRF(FIX),BINARY(BYTE)".

4.2.2 Files Included

- **mf.kbd**

If you use ACCEPT statement with SCREEN SECTION and you want ACCEPT statement input to be terminated by function keys F10, ESCAPE and you want function keys to generate the same codes as in MF, then you need to specify to FSC run-time keyboard mapping contained in file **mf.kbd**.

When you start EXE file of your application compiled with FSC, RunTime Environment Setup dialogue appears. Put string **@CBR_SCR_KEYDEFFILE=mf.kbd** into Environment Variables Information field and push Set and Save buttons. This informs FSC runtime that keyboard mapping from **mf.kbd** should be used with this application.

- **translio.cbl**

If your program uses File IO status codes and the codes that you use are different for your original compiler and FSC Cobol97 then use option **-iostatus-conv**, so that for every File IO statement **mf2fsc** generates call to TRANSLIO routine that is contained in enclosed file **translio.cbl**.

This call is added after the File IO statement and its goal is to convert File IO status from your system convention to FSC convention.

You may want to customize this file with IO status conversion that are appropriate for your original system.

The included version of file translates MS Cobol status codes to FSC Cobol status codes.

4.3 ICobol to Fujitsu Cobol converter

i2fsc converts ICobol to Fujitsu Cobol.

By default the converted file has extension ".fsc".

The complete list of conversions performed can be found in the User's Guide.

4.3.1 Options

- **-add-end-stmts**

Add END-IF, END-SEARCH, END-EVALUATE, END-PERFORM closing statements.

Most Cobol compilers can guess where the composite statement should end, so you are not required to have the closing END-* statements.

However it is considered a good programming style to have all composite statements terminating END-* statements coded explicitly. Transformation invoked by this option automatically adds END-* closing statements.

Default: Off.

- **-pic9toX-only**

Specify that the only transformation to be performed is conversion of binary data items with PIC 9(n) to PIC X(approx n/2).

Used to convert copy libraries.

Default: Off.

- **-ebcdic-sign-values**

Switch the use of the ICOBOL sign byte values for the converted program.

Used to emulate the following ICobol feature: for signed display numeric items with included signs (i.e. not SIGN SEPARATE) ICOBOL sets the sign-carrying byte to the ASCII value for the character that would have been set in an IBM mainframe EBCDIC system.

Default: On.

- **-initialize-tally-item**

Explicitly initialize tally-items to zero before INSPECT statements.

ICOBOL (compiled for ANSI 74 - common with HDS) initializes tally-item to zero (contrary to the ANSI standard). Fujitsu COBOL complies with the standard and does not initialize tally-item.

Default: On.

- **-printer-queue-name**

Specify name of the data item that will be used to store the printer name.

Used to convert PRINTER files with PRINTER-PATH specified.

Default: "PRINTQUE-PRINTER-NAME".

- **-nx-xd-substitute**

Define an extension to be substituted for ".nx" and ".xd" in ##F searches.

Default: "dat".

- **-nxxd-warning**

Issue warning when .NX or .XD literals are found.

Default: On.

- **-print-file-lengths**

List option: `-print-file-lengths=[print-file-name,length-switch-name,...]`

This will indicate that the record length for `print-file-name` is contained in the data item `length-switch-name`.

`Print-file-name` should be the name of the file in the COBOL program. `Length-switch-name` should be a numeric item (at least PIC 9(3)) within the program.

Default: "PRINT-RECORD-LENGTH" variable to be used for all files.

4.4 Year 2000 Window Fix converter

`y2k-fix` fixes Year 2000 problem using Windowing approach.

Y2K-FIX adds windowing logic to a Cobol program. It parses the specified Cobol source file and produces a file with suffix ".yfx" that contains the fixed program.

Specifically, if a comparison operation `>`, `>=`, `<`, `<=` arguments are such that one of them is data-item that contains date, then both arguments of this comparison operation are replaced with temporary data-items, which are assigned Y2K-expanded date values.

Statements that compute expanded values are inserted before the statement that contains this comparison operation.

By default the converted file has extension ".yfx".

4.4.1 Options

- **-date-pic**

If you specify this option, all data-items whose PICTURE is equal to the specified picture string, will be considered data-items that contain date.

Example: `y2k-fix -date-pic=9(6) y2k.cbl`

Here file `y2k.cbl` is parsed and analyzed. File `y2k.yfx` is created that contains the fixed program.

Note: in many shells you need to enclose `9(6)` in quotes like this: `"9(6)"`. This is because (and) have special meaning in many shells,, especially in UNIX shells.

Default: none.

- **-date-file**

If you specify this option, all data-items whose full names are listed in the specified file, are considered to be data-items that contain dates.

Example: put a list of date data-items into file `y2k.dates`:

```
TRAN-SETTL-DATE OF TRAN-DATA OF TRAN-POST OF TRAN
TRAN-TRADE-DATE OF TRAN-DATA OF TRAN-POST OF TRAN
TRAN-REG-DATE OF TRAN-DATA OF TRAN-POST OF TRAN
W-START-DATE OF TRAN
```

Command line: `y2k-fix -date-file=y2k.dates y2k.cbl`

Default: none.

Chapter 5

Cobol Data File Converters

5.1 Cobol File Descriptor Extractor

`cb12fdd` extracts File Descriptor data from a Cobol program and puts it into RDD and FDD files. These files contain machine-readable record descriptors that can be used by our File Access Library to read Cobol VSAM data files generated by many Cobol systems.

RDD file (stands for Record Descriptor Data) fully describes format of the Cobol file record. This data is taken from the record descriptor contained in FD for a file.

FDD file (stands for File Descriptor Data) fully describes file data such as file organization, access method and so on. This data is taken from FD and SELECT statements for a file.

RDD and FDD files are text files but they are easily readable by a computer program. FDD file contains data taken both from SELECT and FD statements for a file.

`cb12fdd` creates and writes RDD and FDD files for each FD in a source program. For each FD it creates one FDD and one or more RDD files. One RDD file is created for every 01 record layout present in FD for a file. Names of these files consist of file name taken from FD and `.rdd` or `.fdd` extension.

`cb12fdd` accepts most of the standard command-line parsing options.

5.1.1 Options

- `-expand-tables`

Create a separate DATA line for each element of Cobol table in a record.

Default: On.

- `-out-file`

If this options is set, output all RDD descriptors to the file whose name is equal to the file name base specified here plus `".RDD"` extension, and output all FDD descriptors to the file whose name is equal to the file name base specified here plus `".FDD"` extension.

If this option is not set, output FD/DD descriptor for every FD/DD specified below to files whose names are equal to FD/DD name plus `".RDD"` or `".FDD"` extension.

Default: not set.

- `-for-files`

If this option is set to list of files, output file descriptors for files specified in this list.

If this options is not set and `-for-files` options is not set, output file descriptors for every file declared in a program.

Default: not set.

- `-for-data-tems`

If this option is set to list of names, output record descriptors for data items specified in this list.

Data items are located by their first name. Since there can be many data items with the same file name, we write out data item descriptor to RDD file whose name is derived from the fully qualified name of the data item.

If it is not set, do not output data item descriptors.

Default: not set.

- `-group-items`

If On, descriptors for group (non-elementary) data items will appear in RDD file. If commented, descriptors for group (non-elementary) data items will appear in RDD file, but corresponding DATA strings will be commented out. Remember that group data items cannot be converted to C++ types, they can only be printed in hexadecimal format.

Default: off.

5.1.2 FDD: File Descriptor Data

This document describes format of FDD file.

FDD file describes the Cobol data file.

It contains all data that is needed to read the file.

FDD file usually has *.FDD extension.

FDD file format is simple and
it is easily readable by computer programs.

FDD file consists of lines.

Line can be of any length.

There is no line continuation character,
so lines cannot be broken.

Lines that start with '#' or '*' or '/'
are comment lines and as such they are ignored.

Non-comment lines consist of fields.

Fields are separated by one or more spaces (' ') or TABs ('\t').
Each field consists of non-space characters.

First field of a line is a keyword that defines

the interpretation of the remainder of the line.
Every keyword starts a command.
All commands take exactly one line.

The following commands are available:

* FILE <cobol-file-name>

Specifies file-name as it appears in the Cobol
program in FD and SELECT statements.
This is here for informational purposes only.

* RDDFILE <file-name>

Specifies name of the RDD file that describes structure of the file record.
If <file-name> contains spaces or special characters,
it must be enclosed in ' or " quotes.
If <file-name> is not absolute, then it is relative
to the directory of FDD file that contains this RDDFILE command.
You may have several RDD files defined for one data record.

* DATAFILE <file-name>

Specifies name of the Cobol data file that contains the actual data.
If <file-name> contains spaces or special characters,
it must be enclosed in ' or " quotes.
If <file-name> is not absolute, then it is relative
to the directory of FDD file that contains this DATAFILE command.
Exactly one DATAFILE command must be present.

* ORGANIZATION <org-type>

Specifies data file organization.
<org-type> can be one of the following:

SEQUENTIAL	ORGANIZATION IS SEQUENTIAL
LINESEQUENTIAL	ORGANIZATION IS LINE SEQUENTIAL
BINARYSEQUENTIAL	ORGANIZATION IS BINARY SEQUENTIAL
INDEXED	ORGANIZATION IS INDEXED
RELATIVE	ORGANIZATION IS RELATIVE

* ACCESSMODE <am-type>

Specifies data file access mode.
<am-type> can be one of the following:

SEQUENTIAL	ACCESS MODE IS SEQUENTIAL
DYNAMIC	ACCESS MODE IS DYNAMIC
RANDOM	ACCESS MODE IS RANDOM

* FILEFORMAT <format-type>

Specifies vendor-specific physical format of the data file.
The following physical formats are available:
DEFAULTFILEFORMAT default format for the program that uses FDD file.

	If no file format is specified, then DEFAULTFILEFORMAT is assumed.
FSC	Fujitsu data file
MF	Micro Focus data file
ACU	AcuCobol data file
RM	Ryan McFarland data file

5.1.3 RDD: Record Descriptor Data

This document describes format of RDD file.

RDD file describes the exact format of
the Cobol data record (01 or FD record description).

RDD file usually has *.RDD extension.

RDD file format is simple and
it is easily readable by computer programs.

RDD file consists of lines.
Line can be of any length.
There is no line continuation character,
so lines cannot be broken.

Lines that start with '#' or '*' or '/'
are comment lines and as such they are ignored.

Non-comment lines consist of fields.
Fields are separated by one or more spaces (' ') or TABs ('\t').
Each field consists of non-space characters.

If a particular field has no value (is empty),
it is represented by '@' character.

First field of a line is a keyword that defines
the interpretation of the remainder of the line.

The following line types are available:

* DATA line defines data item.
This is the most popular line in RDD file.

```
DATA <log-level> <phys-level> <name> <category>
    <usage> <sign> <occurs-from> <occurs-to>
    <offset> <bit-offset> <length> <picture> [<date-picture>]
(all these <> items appear on one line in RDD file)
```

<log-level> is logical level of the item in the hierarchy of data items.
Starts at 0 and is incremented by 1. Can be 0, 1, 2, 3, 4, ...

<phys-level> is physical level of the item as specified in the Cobol program.
Has exactly 2 numeric decimal characters.
Can be: 01, 02, ..., 05, 06, ..., 10, ..., 49.

<name> is the name of the data item.
Conforms to Cobol rules for user-defined names.
If name is empty (@ character), then
this is FILLER data item.

<category> is effective category of the data item.
It can have on the following values:

G	group item
I	index
P	pointer
PP	procedure pointer
A	alphabetic
N	numeric
AN	alphanumeric
AE	alphanumeric edited
NE	numeric edited
IF	internal float
EF	external float
B	bit
J	national (usually Japanese)
JE	national edited

<usage> is effective USAGE clause of the data item.
Effective usage means: If this item has no usage clause, then
the usage clause of the nearest ancestor of the data item is used.
If none of ancestors has usage clause, then USAGE DISPLAY is used.
Usage can be one of the following:

B	BIT (FSC)
C	BINARY (COMP)
CR	RM COMP (DISPLAY-like)
D	DISPLAY
DM	DISPLAY by MicroFocus
DF	DISPLAY by Fujitsu
DR	DISPLAY by RyanMcFarland
D1	DISPLAY-1 (DBCS)
I	INDEX
PT	POINTER
PP	PROCEDURE-POINTER
PD	COMP-3, PACKED-DECIMAL

0	COMP-0 (MS: same as COMP)
1	COMP-1 (internal float)
1R	COMP-1 by RM (binary)
2	COMP-2 (internal float)
4	COMP-4 (same as BINARY)
5	COMP-5 (variation of BINARY)
6R	COMP-6 (RM)
X	COMP-X (variation of BINARY)

On DM, DJ and DR: D (generic DISPLAY) will work in place of DM, DJ, and DR most of the time and you do not need to specify DM, DJ or DR.

However, if you link fields in Data2Cr, then the actual DM, DJ or DR specifiers must be used.

cbl2fdd generates these specifiers based on the value of -lang option.

<sign> is effective SIGN clause of the data item.

It can be one of the following:

L	SIGN IS LEADING
LS	SIGN IS LEADING SEPARATE
T	SIGN IS TRAILING
TS	SIGN IS TRAILING SEPARATE
@	no sign clause

<occurs-from> <occurs-to> describes the OCCURS clause of the data item:

@	@	no OCCURS clause
<n1>	@	OCCURS <n1> TIMES
<n1>	<n2>	OCCURS <n1> TO <n2> TIMES DEPENDING ON ...

<offset> is decimal number that encodes
byte offset of this data item from the start of the record.
Can be 0, 1, 2, and up.

<bit-offset> is decimal number that encodes
bit offset of this data item from the start of the byte
designated in <offset>. Can be not 0 only for BIT items.
Thus <full-bit-offset> = <offset>*8 + <bit-offset>.

<length> is decimal number that encodes
length of this data item in bytes.

For BIT items <length> is is also in bytes.
The length of BIT item in bits is equal
to the length of unrolled picture string
of the item.

<picture> is PICTURE clause character string.

<date-picture>, if it is present, tells the systems that this data item contains date and/or time and it gives the format of date/time presentation.

Using this format, the system parses the date/time data item and stores it internally as a date/time item, so that at output time date/time-specific formatting can be applied to this data item.

<date-picture> can be present only in numeric and numeric-edited data items. The number that represents date/time is parsed by the system according to the format specified in <date-picture>.

<date-picture> can contain the following substrings:

YY 2-digit year
YYYY 4-digit year
MM 2-digit month: ' 1', '01', ' 2', '02', ..., '11', '12'
DD 2-digit day: ' 1', '01', ' 2', '02', ..., '30', '31'
HH 2-digit hour: '00', ' 0', ' ', ' 1', '01', ..., '23'
NN 2-digit minute: '00', '01', ..., '59'
SS 2-digit second: '00', '01', ..., '59'
B Unused position

All characters used for separating different time components ('/' ':' ',' '.') are removed from the the data item before analyzing it using this template.

That is, if you have numeric-edited data item which stores date time as "98/12/31 23:59:59", then it is turned into numeric data item 981231235959 you should use <date-picture> YYMMDDHHNNSS to recognize this number as date-time.

* DECIMALPOINT line defines a character used to represent decimal point for numeric edited data items.

DECIMALPOINT <COMMA | PERIOD>

If this line is omitted, "DECIMALPOINT PERIOD" assumed.

* CURRENCYCHAR line defines a character used as currency symbol.

CURRENCYCHAR <char>

If this line is omitted, CURRENCYCHAR "\$" assumed.

* ALPHABET line defines alphabet used for file. It may be EBCDIC or ASCII. If this line is omitted, assumed alphabet is ASCII.

* APPLIED_TO may be optionally specified in ALPHABET line after alphabet definition. Valid values are ALL and DISPLAY_ONLY. If "APPLIED_TO ALL" specified, all data read from file should be converted from file's alphabet (some compilers like MicroFocus allow CODE-SET keyword for non-DISPLAY data items). "APPLIED_TO DISPLAY_ONLY" (the default) means that alphabet conversion must be applied to data items whose usage is DISPLAY.

* RECORDLENGTH line defines fixed (minimal) record length.

RECORDLENGTH <nReclen>

<nReclen> is fixed record length.

* MAXRECORDLENGTH line defines maximal record length.

Maximal record length is more than fixed record length in case of variable record length (OCCURS..DEPENDING ON present in record definition).

MAXRECORDLENGTH <nMaxreclen>

<nMaxreclen> is maximal record length.

Real record length lies between nReclen and nMaxreclen.

5.2 Cobol Data File Format Guess Program

`data-guess` guesses both file format and record layout for Cobol data files that it recognizes.

To see a list of data file formats that the program recognizes, start it without any arguments.

`data-guess` generates FDD and RDD files for a given Cobol data file, if it can recognize the file format.

Please note that DataGuess cannot fully recover the RDD file, it only gives you a good approximation of the record layout that you can later adjust manually.

Specifically, data items in the generated record layout have generated names that are not meaningful; several neighboring data items of similar type will be merged into one big data item.

5.2.1 Options

- -help

Print list of command-line options.

Default: Off.

- -out-rdd

Write generated RDD to this file.

Default: console.

- **-out-fdd**

Write generated FDD to this file.

Default: console.

- **-format-hint**

This is a file format hint that helps to distinguish between close formats. For instance, record storage and indexing techniques used in FSC and RM Cobol are pretty much the same, while data item representation format is different. Providing hint in this case helps DataGuess with breaking the record into individual fields.

Default: no hints.

5.3 Cobol Data File To Flat File Converter

`data2flat` converts Cobol data file (sometimes also called VSAM or ISAM file) to flat data file.

In flat file every record occupies exactly one line and data items on the line are separated by separator character (usually comma). Flat file contains text representation of all data items in a file.

`data2flat` converts Cobol data files, not Cobol programs, so it does not accept standard Cobol parser and pretty-printer options. All accepted options are listed below.

5.3.1 Options

- **-help**

Print list of command-line options.

Default: Off.

- **-v**

Print the current version of the tool.

Default: Off.

- **-progress**

Print conversion progress message every 250 records.

Default: Off.

- **-silent**

Suppress runtime messages (file openings, etc.).

Default: Off.

- **-show-descr**

Display the loaded FDD and RDD file descriptor.

Default: Off.

- **-out-file**
Write flat data to this file.
Default: input file with suffix changed to `.flt`.
- **-separator**
Separate fields in a flat file record using this string.
Default: `","` (comma).
- **-quote**
Character used to quote character data items in the flat file. If quote character itself is a part of the character data item, double the quote character.
Default: none.
- **-max-recs**
Number of records to convert. If `-max-recs=0`, then convert all records.
Default: 0 (All records).
- **-dump-items**
List of fields to be dumped in the format specified in `-dump-format`, up to 25. By default, data items specified in `-dump-items` and group data items are dumped in hexadecimal format.
Default: none.
- **-dump-format**
Specify the dump format used for data items specified in `-dump-items` and for group data items. Can be one of the following:

<code>hex</code>	Hexadecimal
<code>alpha</code>	Text (alphanumeric)

Default: `hex`.
- **-europe-date**
Print dates as `DD.MM.YYYY` (Europe format). By default, dates are printed as `MM/DD/YYYY` (American format).
Default: Off.

5.4 Cobol Data File To DBase IV DBF File Converter

`data2dbf` converts Cobol data file to DBF data file. DBF is a file format used by dBase IV database. You can import DBF files into Microsoft Excel and other spreadsheet and reporting software.

`data2dbf` converts Cobol data files, not Cobol programs, so it does not accept standard Cobol parser and pretty-printer options. All accepted options are listed below.

5.4.1 Options

- -help
Print list of command-line options.
Default: Off.
- -v
Print the current version of the tool.
Default: Off.
- -progress
Print conversion progress message every 250 records.
Default: Off.
- -show-descr
Display the loaded FDD and RDD file descriptor.
Default: Off.
- -out-file
Write flat data to this file.
Default: input file with suffix changed to .flt.

5.5 Crystal Reports Cobol Data Reader DLLs

Cobol data files driver for Crystal Reports (CR)
=====

Features

1. Reads Cobol data files produced by FSC and MF Cobol programs.
2. Links several data files in one report using indexes.
3. Links from DBF files to Cobol data files and,
in some cases, from Cobol data files to DBF files.

FDD and RDD files: File Descriptors

Files with *.FDD and *.RDD extension contain
all the data that Data2Cr needs from a Cobol program
to read the actual Cobol data file.

Since Cobol data file does not have record descriptor data embedded,
we have to go thru extra layer of FDD and RDD files that connect
physical file to record description.

FDD file contains data from SELECT statement for a file,
such as file organization and access method.
FDD file also contains a link to the actual Cobol data file.

RDD file contains data from FD statement for a file.
Specifically, it contains record layout.

FDD and RDD file formats are described in Cbl2Fdd manual.

More on this at <http://www.siber.com/sct/datafile/>

Cbl2Fdd Program

Siber Systems supplies a separate program called CBL2FDD.
This program automatically generates FDD and RDD files
from Cobol programs in many dialects, including FSC Cobol.

Please note that Data2Cr installer does NOT install CBL2FDD.
You need to install CBL2FDD as a separate product.

Cbl2Fdd is distributed as one WIN32 executable file
that self-extracts and then installs Cbl2Fdd.

Please see CBL2FDD distribution for details.

Installation

* Just run the executable distribution file that is named
fscdata2cr-3-5-6-fsc.exe, mfdata2cr-3-5-6-sib.exe or similar to that.

* When installer asks, whether you want to
copy file "p2bfsc.dll" ("p2bmf.dll" for MF reader)
to \$systemroot\Crystal, answer "Yes".
Our DLL becomes useable by
Crystal Reports only after being copied
to \$systemroot\Crystal directory.

DLL conflicts

* Standard P2BBDE.DLL that comes with CR recognizes

any file that is longer than 31 bytes as its own.
It recognizes FDD files too (if they are longer than 31 bytes) and
therefore it prevents our P2BFSC.DLL (P2BMF.DLL) from recognizing FDD file.

When P2BBDE.DLL takes over, you would see
only "FIELD1" in the list of fields and nothing else.

* In Crystal Reposrts ver 6 and later there is a bug
in P2BBTRV.DLL. It sometiems results in error message
"Could not open pdbbtrv.dll. Please check its configuration"
when you try to open FDD file. Whether the bug appears
or not depends on physical order of DLLs in directory.

We recommend 2 ways to fix it:

- Method 1 (preferred method implemented by Data2Cr installer):

Remove P2BBDE.DLL and P2BBTRV.DLL from \$systemroot\Crystal.
Save it in some other directory, say, \$systemroot\CrystalDisabled,
so that when you need its functionality, you can move
it back to \$systemroot\Crystal.

Shortcomings: Paradox, Btrieve and other databases that
are read by P2BBDE.DLL and P2BBTRV.DLL will not be
loadable to CR.

Advantages: simple and makes Cobol stuff work.

Crystal Reports 4.5 has no P2BBDE.DLL,
so there is no need to remove it.

- Method 2:

This method is more complicated but it allows you
to use both Cobol and BDE/Btrieve file reading DLLs.

<<< DO NOT USE THIS METHOD UNLESS YOU REALLY KNOW WHAT YOU ARE DOING <<< start

- a) if you used Data2Cr installer that performed Method 1,
then you would need to move P2BBDE.DLL and P2BBTRV.DLL
back to \$systemroot\Crystal.
- b) If you installed CR version 6 or later then you have
defective P2BBTRV.DLL. Replace it with P2BBTRV.DLL
from CR versions before 6 or simply disbale it by

moving it to \$systemroot\CrystalDisabled.

- c) Rename P2BxBSE.DLL to P2BDBSE.DLL in \$systemroot\Crystal.
This will force CR database manager to look for extra DLLs in \$systemroot\Crystal.
- d) For every FDD file create file with the same base name, but different extension. We use FDL extension.
For instance, for file TEST.FDD create file TEST.FDL.
FDL files must be shorter than 31 bytes, so let us make it just one byte long by putting in one empty line.
Contents of FDL file do not matter.

FDL file will not be recognized by P2BBDE.DLL because they are less than 32 bytes long, and then our DLL can pick them up. It will translate FDL extension to FDD extension and then it will load FDD file instead.
For this example TEST.FDD file will be loaded instead of TEST.FDL.

>>> DO NOT USE THIS METHOD UNLESS YOU REALLY KNOW WHAT YOU ARE DOING >>> end

How To Open Cobol data files in Crystal Reports

When adding a Data Source to report select Data File.

Ask Open Dialogue to list "All Files".

Select a file with *.FDD extension that has format as described above
(select *.FDL file if you are using method 2 of fighting DLL conflicts).

If filenames/paths in FDD file are correct,
data from RDD and VSAM files will be loaded and
you will see fields of the virtual database
created by Data2Cr DLL from your Cobol data file.

Example and Test

Start Seagate Crystal Reports Designer.
Create new Standard report by selecting File.New.

Click "DataFile" button. List files of type: All files.
Select file-fsc.fdd (for FSC) or file-mf.fdd (for MF)

file in Data2Cr distribution directory
which usually is D:\Program Files\Siber Systems\fscdata2cr or mfddata2cr.
Once the file is parsed, click Done to close the dialogue.

Click Field tab. Now you should see all the fields from
the Cobol record listed: OUT-KEY, OUT-S-COMP-3, ..., FS-DISPLAY.
Click AddAll to add all of them to the report.

Finally click PreviewReport to see the report
generated from the Cobol data file.

OCCURS issues

Data item: name OCCURS n TIMES PIC p USAGE u
is presented both:

- as a concatenation of n items that has "PIC p(n) USAGE u" format, and
- as n items with names name_1, name_2, ..., name_n,
each having "PIC p USAGE u" format.

Data items with USAGE IS BIT

This library can read data items with USAGE IS BIT. Since
CR cannot show numeric data in format other of decimal, we convert
such data to strings. These strings can be used as index pattern
for file linking; they cannot be used as index keys.

Fixing Y2K

For Crystal Reports, all dates must come in 4-digits-year format. This
is not always so. To fix it, this library will automatically 2-digits-year
dates to 4-digits-year dates. Rule for this conversion is following:
if year is less than 40, it will be converted to 20YY, otherwise, to 19YY.

Known issues in FSC F3BIFCFA library

We use F3BIFCFA library from Fujitsu to access
Fujitsu COBOL data file. This library has bugs/features
that may adversely affect functionality of Data2Cr.

1. If you install Sibver Systems FscData2Cr on a computer

that has *no* Fujitsu compiler installed then you need to add to your PATH FscData2Cr directory that contains F3BIFCFA.DLL and F3BIFRM.DLL used in reading data files (usually \Program Files\Siber Systems\fscdata2cr). If you do not do it, then you will get the following error: "Configuration Error. Library pdbfsc.dll cannot be opened. Please check its configuration".

2. Sometimes it is not possible to read file for which access mode is LINE SEQUENTIAL and which contains numeric data with USAGE other than DISPLAY. If \n or \r bytes are contained in numeric field (it is possible for USAGE BINARY/COMP-* data items), F3BIFCFA treats such bytes as the end of record.

This is fixed by changing access mode to RECORD SEQUENTIAL and adding explicit FILLER field to the end of record to cover CR/LF.

3. Since F3BIFCFA has no way of accepting information on USAGES of fields, it treats all fields as strings even when using these fields for indexing.

For instance, if COMP-1 field for which ordering dictated by string representation of value image is different from correct ordering dictated by Cobol field interpretation as a number is used as a part of index in INDEXED file, then one cannot retrieve records using such indices.

The following data format combinations cause misordering by F3BIFCFA:

- USAGE is COMP-1, COMP-2, or COMP-5
- SIGN IS TRAILING / SIGN IS TRAILING SEPARATE / SIGN IS LEADING.

This problem prevents linking files that use such record keys.

Valid combinations are:

USAGE IS DISPLAY. SIGN IS LEADING SEPARATE.

USAGE IS BINARY. (positive values only)

USAGE IS COMP. (positive values only)

USAGE IS COMP-3. SIGN IS LEADING SEPARATE.

USAGE IS COMP-3. SIGN IS LEADING. (positive values only).

Data2Cr does not report these problems, it just calls F3BIFCFA which results in records not being found.

FAQ

Q: How do I handle keys that are group items?

A: Group data items (the ones that are not elementary data items) cannot be used as indices.
 However, you can use a concatenation of elementary items in a group as a key.

Say, you have the following program:

```

000070      SELECT OPTIONAL SCA1060 ASSIGN TO WS-SCA1060
000080          ORGANIZATION      IS INDEXED
000090          ACCESS MODE        IS DYNAMIC
000100          RECORD KEY        IS SCA1060-CHAVE
000110          ALTERNATE RECORD KEY IS SCA1060-CHAVE-1
000120                                WITH DUPLICATES
000110          ALTERNATE RECORD KEY IS SCA1060-CHAVE-2
000120                                WITH DUPLICATES
000110          ALTERNATE RECORD KEY IS SCA1060-CHAVE-3
000120                                WITH DUPLICATES
000130          LOCK MODE IS AUTOMATIC WITH LOCK ON RECORD
000140          FILE STATUS        IS WS-STATUS.

FD      SCA1060 IS GLOBAL.
01      SCA1060-REG.
    03  SCA1060-CHAVE.    // Cannot use this; instead, use ----+
    05  SCA1060-CODIGO    PIC 9(006). <-----+
    03  SCA1060-CHAVE-1.
    05  SCA1060-NOME      PIC X(040).
    03  SCA1060-CHAVE-2. // Cannot use this; instead, use all of --+
    05  SCA1060-CODFEITO  PIC 9(005). <---+ |
    05  SCA1060-NOME-2    PIC X(040). <---+-----+
    05  SCA1060-CODIGO-2  PIC 9(006). <---+
    03  SCA1060-NOMEFEITO PIC X(040).
    03  SCA1060-CHAVE-3.
    05  SCA1060-CODTPVARA      PIC 9(004).
    05  SCA1060-NOME-3        PIC X(040).
    05  SCA1060-CODIGO-3      PIC 9(006).
    03  SCA1060-NOMETPVARA    PIC X(040).
  
```

Here all record keys are group items.
 Crystal will not show content of such items, since their format is unknown.

So to link this file as a slave file
 you should use SCA1060-CODIGO instead of SCA1060-CHAVE;
 or, you can use SCA1060-CODFEITO, SCA1060-NOME-1, SCA1060-CODIGO-2
 instead of SCA1060-CHAVE2.

--

Copyright 1998-2000 by Siber Systems.
All rights reserved.

Chapter 6

Legal

6.1 Paid-For and Shareware Tool License

SIBER SYSTEMS INC
END USER SOFTWARE LICENSE AGREEMENT

IMPORTANT: You should carefully read this legal agreement before installing this package. By installing this software, you accept all the terms and conditions of this agreement and agree to abide by them. If these terms and conditions are not acceptable to you, do NOT continue to install this software and return it to your distributor and your money will be refunded.

SIBER SYSTEMS PROGRAMS LICENSE TERMS

Siber Systems Inc ("Siber") grants you ("Customer") a license to use the enclosed software and documentation ("Programs") as indicated below.

LICENSE: Customer shall have the right to use the Programs, either
(a) to the extent specified in an ordering document or Program Use Certificate distributed to Customer by Siber Systems or its distributor, or
(b) if not specified, for a single user on a single computer.

Customer may use the Programs solely for its own internal data processing operations. Customer may make one copy of each licensed Program for backup; rights to make additional copies, if any, may be specified in an ordering document or Program Use Certificate. No other copies shall be made without Siber Systems prior written consent.

Customer shall not:

(a) remove any product identification, copyright notices,

or other notices or proprietary restrictions from Programs;
(b) use Programs for commercial timesharing, rental,
or service bureau use;
(c) transfer, sell, assign or otherwise convey Programs to
another party without Siber Systems prior written consent;
(d) cause or permit reverse engineering, disassembly,
or decompilation of Programs; or
(e) disclose results of any benchmark tests of any Program
to any third party without Siber Systems prior written approval.

All Program transfers are subject to Siber Systems transfer
policies and fees. Customer shall have the right to use only
the Programs in this package that are specified in an ordering
document or Program Use Certificate.

COPYRIGHT/OWNERSHIP OF PROGRAMS: Programs are the proprietary
products of Siber Systems and its licensors and are protected
by copyright and other intellectual property laws.
Customer acquires only the right to use Programs and
does not acquire any rights, express or implied, in
Programs or media containing Programs other than
those specified in this License.
Siber Systems, or its licensor, shall at all times retain
all rights, title, interest, including intellectual
property rights, in Programs and media.

LIMITED WARRANTIES/EXCLUSIVE REMEDIES: Siber Systems warrants
that for thirty (30) days from date of delivery to Customer:
(a) enclosed media is free of defects in materials and
workmanship under normal use; and
(b) unmodified Programs will substantially perform functions
described in documentation provided by Siber Systems when
operated on the designated computer and operating system.

Siber Systems does NOT warrant that:
Programs will meet Customer's requirements,
Programs will operate in combinations Customer may select for use,
operation of Programs will be uninterrupted or error-free, or
all Program errors will be corrected.
These warranties are exclusive and in lieu of all other
warranties, whether express or implied, including implied warranties
of merchantability or fitness for a particular purpose.

If Customer reports an error in a Program within
the thirty (30) day period, Siber Systems shall, at its option,
correct the error, provide Customer with a reasonable procedure
to circumvent the error, or, upon return of Programs

to Siber Systems by Customer, refund the license fees. Siber Systems will replace any defective media without charge if it is returned to Siber within the thirty (30) day warranty period. These are Customer's sole and exclusive remedies for any breach of warranty. This limited warranty gives you specific legal rights. You may have others, which vary from state to state.

LIMITATION OF LIABILITY: NEITHER SIBER SYSTEMS NOR ANY OF ITS LICENSORS SHALL BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES, OR DAMAGES FOR LOSS OF PROFITS, REVENUE, DATA OR DATA USE, INCURRED BY CUSTOMER OR ANY THIRD PARTY, WHETHER IN AN ACTION IN CONTRACT OR TORT, EVEN IF SIBER SYSTEMS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SIBER SYSTEMS AND ITS LICENSORS' LIABILITY FOR DAMAGES HEREUNDER SHALL IN NO EVENT EXCEED THE FEES PAID BY CUSTOMER FOR THIS LICENSE.

TECHNICAL SUPPORT: Siber Systems will provide limited technical support to purchasers of this Program for thirty (30) days after its purchase and receipt of payment for the Program. Technical support can be obtained via e-mail at support@siber.com.

Siber Systems acknowledges all trademarks found in this license, in the software package, and in the documentation.

RESTRICTED RIGHTS: Programs delivered to the U.S. Defense Dept. are delivered with Restricted Rights and the following applies: "Restricted Rights Legend: Use, duplication or disclosure by the Government is subject to restrictions as currently set forth in subparagraph (c)(1)(ii) the Rights in Technical Data and Computer Software clause at 252.227-7013 (or any successor regulation). Siber Systems Inc., 2902 Rock Manor Ct, Herndon, VA 20171".

Programs delivered to a U.S. Government Agency not within the Defense, Dept. are delivered with "Restricted Rights" as defined in Commercial Computer Software - Restricted Rights at FAR 52.227-19.

Customer shall comply fully with all laws and regulations of the United States and other countries (Export Laws) to assure that neither the Programs, nor any direct products thereof are (1) exported, directly or indirectly, in violation of Export Laws, or (2) are used for any purpose prohibited by Export Laws, including, without limitation, nuclear, chemical, or biological weapons proliferation. This License and all related actions shall be governed by Virginia law.

Siber Systems may audit Customer's use of the Programs. All terms of any Customer purchase order or other Customer ordering document shall be superseded by this License.

6.2 Trial Tool License

SIBER SYSTEMS INC

EVALUATION SOFTWARE LICENSE AGREEMENT

IMPORTANT: YOU SHOULD CAREFULLY READ THIS LEGAL AGREEMENT BEFORE INSTALLING THIS PACKAGE. BY INSTALLING THIS SOFTWARE, YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT AND AGREE TO ABIDE BY THEM. IF THESE TERMS AND CONDITIONS ARE NOT ACCEPTABLE TO YOU, DO NOT CONTINUE TO INSTALL THIS SOFTWARE.

SIBER SYSTEMS PROGRAM LICENSE TERMS

Siber Systems Inc ("SIBER") grants you ("Customer") a license to use the enclosed software and documentation ("Programs") as indicated below.

LICENSE: Customer shall have the limited right to use the Programs for evaluation purposes for Thirty (30) days. Programs are trial version and applications converted by Programs cannot be distributed commercially. Customer can use the applications only to evaluate Programs.

Customer shall not:

- (a) remove any product identification, copyright notices, or other notices or proprietary restrictions from Programs;
 - (b) use Programs for commercial purposes or for timesharing, rental, or service bureau use;
 - (c) transfer, sell, assign or otherwise convey Programs to another party without Siber Systems prior written consent;
 - (d) cause or permit reverse engineering, disassembly, or decompilation of Programs; or
 - (e) disclose results of any benchmark tests of any Program to any third party without Siber Systems prior written approval.
- Any Program transfers permitted by Siber Systems hereunder are subject to Siber Systems transfer policies and fees.

COPYRIGHT/OWNERSHIP OF PROGRAMS: Programs are the proprietary products of Siber Systems and its licensors and are protected by copyright and other intellectual property laws. Customer acquires only the right to use Programs for evaluation purposes

and does not acquire any rights, express or implied, in Programs or media containing Programs other than those specified in this License. Siber Systems, or its licensor, shall at all times retain all rights, title, interest, including intellectual property rights, in Programs and media.

NO WARRANTIES: THIS IS AN EVALUATION COPY OF THE PROGRAMS. SIBER SYSTEMS IS PROVIDING THE PROGRAMS AS IS WITHOUT ANY WARRANTY OF ANY KIND AND FURTHER DISCLAIMS ALL IMPLIED AND STATUTORY WARRANTIES INCLUDING THE WARRANTIES OF TITLE, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. EVALUATION SOFTWARE SHOULD NOT BE USED FOR PRODUCTION USE OR WITH LIVE DATA.

LIMITATION OF LIABILITY: NEITHER SIBER SYSTEMS NOR ANY OF ITS LICENSORS SHALL BE LIABLE FOR ANY INDIRECT, INCIDENTAL, EXEMPLARY, PUNITIVE, SPECIAL OR CONSEQUENTIAL DAMAGES, OR DAMAGES FOR LOSS OF GOODWILL, PROFITS, REVENUE, DATA OR DATA USE, INCURRED BY CUSTOMER OR ANY THIRD PARTY, WHETHER IN AN ACTION, IN CONTRACT OR TORT, EVEN IF SIBER SYSTEMS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

RESTRICTED RIGHTS: Programs delivered to the U.S. Defense Dept. are delivered with Restricted Rights and the following applies: "Restricted Rights Legend: Use, duplication or disclosure by the Government is subject to restrictions as currently set forth in subparagraph (c)(1)(ii) the Rights in Technical Data and Computer Software clause at 252.227-7013 (or any successor regulation). Siber Systems Inc., 2902 Rock Manor Ct, Herndon, VA 20171".

Programs delivered to a U.S. Government Agency not within the Defense, Dept. are delivered with "Restricted Rights" as defined in Commercial Computer Software - Restricted Rights at FAR 52.227-19.

COOPERATION: Customer shall promptly report any bugs or other defects to Siber Systems. Customer agrees to participate with Siber Systems in selected marketing activities related to the Programs.

GENERAL: Siber Systems may audit Customer's use of the Programs. This License states the full understanding of the parties regarding Customer's use of the Programs. All terms of any Customer purchase order or other Customer ordering document shall be superseded by this License.