File Tools: Reference and Guide Version 2

CML00043-02

Code Magus Limited (England reg. no. 4024745)
Number 6, 69 Woodstock Road
Oxford, OX2 6EY, United Kingdom
www.codemagus.com

December 15, 2020

Contents

1 Introduction

Code Magus Filetools is a collection of generalised utility programs for manipulating data in a variety of formats. They make the process of copying, massaging, converting, browsing and profiling data much easier than using the standard utilities found on the many diverse platforms they run on. Their execution is consistent between the different utilities and the platforms, which improves user productivity.

All the Filetools tools run on Unix and Unix-based platforms, Windows platforms, z/OS Unix System Services (USS) and Classic z/OS MVS environments. See appendix B on page 45 for examples of how to run these programs under MVS. Examples of invoking the tools from the Unix/Linux/DOS command line are shown at the end of of each section; the commands may be split over multiple lines for formatting in this document, but should be seen as one continuous line. All supporting files (objtypes, copybooks and files) are shown in the appendixes.

All the tools from Version 2 onward use the Code Magus Record Stream I/O (`Recio`) to read and write data files. The library is explained in the `Recio` manual recio: Record Stream I/O Library Version 1 [1]. The `Recio` library allows access to data files via named access methods. These access methods need to be installed and configured for use. The following is a non-exhaustive list of the main access methods:

- Binary is described in the manual:
  binary: Fixed and Variable Length Record Stream Access Method Version 1 [2] and allows binary data to be read and written in either fixed or variable format records.
- Text is described in the manual:
  text: File Access Method Using POSIX Streams Version 1 [7] and allows programs to read and write text based files on any platform for any platform. The platforms are Windows; Unix and Linux; and Unix System Services on z/OS. Each platform recognises records with a specific line ending.
- Edit is described in the manual:
  edit: Recio Edit Access Method Version 1 [9] and allows for automated record editing during either the read or write phase. This is convenient when records are written in some compressed form, but need to be expanded for processing when read.
- MVS is described in the manual:
  MVS: MVS Record Stream Access Method Version 1 [4] and allows the file to be specified in exactly the same syntax as z/OS MVS JCL (Job Control Language)
- Remote is described in the manual:
  remote: Remote Record Stream Access Method Version 1 [5] and allows files to read and written to or from a remote server using any other access method.

- Image is described in the manual:
  image: DB2 Image Copy Reader Access Method Version 1 [8] and allows programs to read DB2 image copies.
- DB2query is described in the manual:
  db2query: Recio DB2 Query Access Method Version 1 [10] and allows programs to read from DB2 using SQL statements.
- DB2DCLGN is described in the manual:
  db2dclgen: Recio DB2 DCL Generator (DCLGN) Access Method Version 1 [11] and allows programs to read the meta-data from DB2 SQL statements.
- Standard is described in the manual:
  standard: Standard Input And Output Using Recio Version 1 [6] and allows records to read from and written to standard in and standard out respectively.
- Directory is described in the manual:
  directory: Directory Record Stream Access Method Version 1 [3] and allows a program to read a directory listing in real time.

All the tools also use either the Code Magus `symbols` [12] library or the Code Magus `Objecttypes` [13] library to process the copybooks in order to generate meta-data structures to and map the data records being manipulated.

Where applicable the tools allow the user to supply a conditional expression that selects a subset of records from the input using the given predicates in each expression. There are numerous built in functions that can be used in these expressions and the expression syntax and built in functions are described in appendix C on page 47. The user can specify a conditional expression using the `--select, -w` option on the command line. See the example for cmlcopy in section 3.4.3 on page 15.

2 Code Magus Print/Unpack `cmlprint`

2.1 Introduction

The `cmlprint` tool reads an input file mapped by an objtypes [13] definition and produces one of a number of different formatted outputs.

2.2 Processing

The data can be filtered by the object types definition and various of the command line parameters and may be encoded in a different character set to that of the running system. From the selected input `cmlprint` produces one of a number of different formatted outputs, namely:

- a formatted print of the meta-data structure of each record and its corresponding data. This is the default output class if none is specified.
- a comma separated file (CSV) ready for loading into a spreadsheet application. This performs the unpack function; `cmlpack` performs the reverse of this function.
- a Code Magus XL export formatted file.
- a plain hex dump of the file.

2.3 Command line parameters

```
Usage: cmlprint <access>(<object>[,<options>]) ...
  -b, --format={structure|csvfile|xlexport|dump}        Format for printing
                                                        the file
  -o, --output-file={stdout|<output file>}              Optional output file
                                                        name
  -c, --charset={ascii|ebcdic}                          Default character set
                                                        for character data
  -e, --endian={big|little}                             Default binary data
                                                        endian
  -q, --skip-input-records={0|<count>}                  Input records to skip
                                                        before processing
  -n, --max-input-records={0|<count>}                   Maximum number of
                                                        input records to
                                                        consider (after skip)
  -j, --max-output-records={0|<count>}                  Maximum number of
                                                        records that should be
                                                        printed
  -d, --bind-binary-mode={word|byte}                    Mode of allocating
                                                        lengths to binary items
  -t, --objtypes={|<objtypes-name>}                     Name of objtypes
                                                        member for buffer
                                                        typing
  -s, --delimiter={,|<delimiter>}                       CSV file field
                                                        delimiter character
```

```
-g, --get-value                                    Extract strings using
                                                   getvalue methods
-f, --type-filter=STRING                           Only select records
                                                   matching given types
-i, --string-long-numbers                          Numeric items longer
                                                   than 10 chars
                                                   formatted as strings
-k, --force-long-strings                           Alphanumeric items
                                                   longer than 10 chars
                                                   formatted as strings
-m, --maximum-records-per-type={0|<max-per-type>}  Maximum records to
                                                   print per object type
-p, --allow-wide-output                            Allow the structured
                                                   output to format wide
-h, --delimit-split-long-values                    Delimit split values
                                                   in narrow structured
                                                   output
-y, --ignore-untype-record                         Skip formatting of
                                                   records without
                                                   matching type
-u, --odo-offset-mode                              Field offsets take
                                                   occurs-depending-on
                                                   clause into account
-w, --select={|<from-where>}                       Select by type and
                                                   optional expression
                                                   over type
-z, --warn-on-bufsel-error                         Continue processing
                                                   on buffer selection
                                                   error
-0, --skip-zero-length-records                     Skip input records that
                                                   have a zero length
-v, --verbose                                      List symbol table and
                                                   parsed copybooks
```

where:

- `-b, --format={structure|csvfile|xlexport|dump}`
  Format for printing the file
- `-o, --output-file={stdout|<output file>}`
  Optional output file name
- `-c, --charset={ascii|ebcdic}`
  Default character set for character data
- `-e, --endian={big|little}`
  Default binary data endian
- `-q, --skip-input-records={0|<count>}`
  Input records to skip before processing
- `-n, --max-input-records={0|<count>}`
  Maximum number of input records to consider (after skip)
- `-j, --max-output-records={0|<count>}`
  Maximum number of records that should be printed
- `-d, --bind-binary-mode={word|byte}`
  Mode of allocating lengths to binary items

- `-t, --objtypes={|<objtypes-name>}`
  Name of objtypes member for buffer typing
- `-s, --delimiter={,|<delimiter>}`
  CSV file field delimiter character
- `-g, --get-value`
  Extract strings using getvalue methods
- `-f, --type-filter=STRING`
  Only select records matching given types
- `-i, --string-long-numbers`
  Numeric items longer than 10 chars formatted as strings
- `-k, --force-long-strings`
  Alphanumeric items longer than 10 chars formatted as strings
- `-m, --maximum-records-per-type={0|<max-per-type>}`
  Maximum records to print per object type
- `-p, --allow-wide-output`
  Allow the structured output to format wide
- `-h, --delimit-split-long-values`
  Delimit split values in narrow structured output
- `-y, --ignore-untype-record`
  Skip formatting of records without matching type
- `-u, --odo-offset-mode`
  Field offsets take occurs-depending-on clause into account
- `-w, --select={|<from-where>}`
  Select by type and optional expression over type
- `-z, --warn-on-bufsel-error`
  If, when trying to determine the type of the current record, an error occurs and this option is specified the record is skipped and processing continues with the next record.
- `-0, --skip-zero-length-records`
  If a the current record read has a zero length then skip it (do not attempt to write the formatted data to the output) and continue processing with the next record. This situation can occur when using `editam`[9] and the edit process encounters an error during record editing (usually expansion of a compressed record on read) and is configured to return a zero length record.
- `-v, --verbose`
  List symbol table and parsed copybooks and produce verbose run time messages

during processing.

2.4 Examples

2.4.1 Print the contents of the example text file.

The various records are formatted as per the type they correspond to in the object types definition file. The header and trailer records do not print any of the data past the key, the data records only print as many EXAMPLE_VARCHAR fields as defined by EXAMPLE_COUNT and there is one record that has an undefined type and is thus printed as a hex dump.

```
cmlprint --objtypes=example.objtypes
      "text(example.txt,mode=r)"

Code Magus Limited Filetools V3.0: build 2017-03-02-16.56.45
[cmlprint] $Id: filetools_cmlprint_eg1.tex,v 1.2 2017/03/06 17:50:37 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].


Start of File = text(example.txt,mode=r).
Using object types = example.objtypes.

Seq = 1, Length = 13
File = text(example.txt,mode=r)
Type = EXAMPLE_RECORDS_HEADER
Title = Filetools Example Record Header

   01 EX_HEAD
      03 RECORD_TYPE = "H"
      03 EX_KEY = 0000
      03 EX_DATE = 20130101

Seq = 2, Length = 17
File = text(example.txt,mode=r)
Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3

   01 EX_REC
      03 RECORD_TYPE = "3"
      03 EX_KEY = 1111
      03 EX_DATA
         05 EX_CHARACTER = "abcde"
         05 EX_COUNT = 05
         05 EX_VC(1) = "a"
         05 EX_VC(2) = "b"
         05 EX_VC(3) = "c"
         05 EX_VC(4) = "d"
         05 EX_VC(5) = "e"

Seq = 3, Length = 13
File = text(example.txt,mode=r)
Type = EXAMPLE_RECORDS_TYPE1
Title = Filetools Example Record Type 1

   01 EX_REC
      03 RECORD_TYPE = "1"
      03 EX_KEY = 1122
```

```
      03 EX_DATA
         05 EX_CHARACTER = "f    "
         05 EX_COUNT = 01
         05 EX_VC(1) = "f"

Seq = 4, Length = 17
File = text(example.txt,mode=r)
Type = EXAMPLE_RECORDS_TYPE1
Title = Filetools Example Record Type 1

   01 EX_REC
      03 RECORD_TYPE = "1"
      03 EX_KEY = 1133
      03 EX_DATA
         05 EX_CHARACTER = "kl  z"
         05 EX_COUNT = 05
         05 EX_VC(1) = "k"
         05 EX_VC(2) = "l"
         05 EX_VC(3) = " "
         05 EX_VC(4) = " "
         05 EX_VC(5) = "z"

Seq = 5, Length = 17
File = text(example.txt,mode=r)
Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3

   01 EX_REC
      03 RECORD_TYPE = "3"
      03 EX_KEY = 1144
      03 EX_DATA
         05 EX_CHARACTER = "pqrst"
         05 EX_COUNT = 05
         05 EX_VC(1) = "p"
         05 EX_VC(2) = "q"
         05 EX_VC(3) = "r"
         05 EX_VC(4) = "s"
         05 EX_VC(5) = "t"

Seq = 6, Length = 17
File = text(example.txt,mode=r)
Last error:
Unable to determine type of buffer:

        00..__..__05..__..__10..__..__15..__..__20..__..__25..__..__30..
  0000: 353131353561626364783035616263646 5
  0000: 5.1.1.5.5.a/b.c.d.x.0.5.a/b.c.d.e.

Seq = 7, Length = 15
File = text(example.txt,mode=r)
Type = EXAMPLE_RECORDS_TYPE1
Title = Filetools Example Record Type 1

   01 EX_REC
      03 RECORD_TYPE = "1"
      03 EX_KEY = 1166
      03 EX_DATA
         05 EX_CHARACTER = "uvw  "
         05 EX_COUNT = 03
         05 EX_VC(1) = "u"
         05 EX_VC(2) = "v"
         05 EX_VC(3) = "w"

Seq = 8, Length = 16
File = text(example.txt,mode=r)
Type = EXAMPLE_RECORDS_TYPE2
```

```
Title = Filetools Example Record Type 2

   01 EX_REC
      03 RECORD_TYPE = "2"
      03 EX_KEY = 1177
      03 EX_DATA
         05 EX_CHARACTER = "zabc "
         05 EX_COUNT = 04
         05 EX_VC(1) = "z"
         05 EX_VC(2) = "a"
         05 EX_VC(3) = "b"
         05 EX_VC(4) = "c"

Seq = 9, Length = 17
File = text(example.txt,mode=r)
Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3

   01 EX_REC
      03 RECORD_TYPE = "3"
      03 EX_KEY = 1188
      03 EX_DATA
         05 EX_CHARACTER = "abcde"
         05 EX_COUNT = 05
         05 EX_VC(1) = "e"
         05 EX_VC(2) = "f"
         05 EX_VC(3) = "g"
         05 EX_VC(4) = "h"
         05 EX_VC(5) = "i"

Seq = 10, Length = 11
File = text(example.txt,mode=r)
Type = EXAMPLE_RECORDS_TRAILER
Title = Filetools Example Record Trailer

   01 EX_TAIL
      03 RECORD_TYPE = "T"
      03 EX_KEY = 9999
      03 EX_RECORDS = 000008

text(example.txt,mode=r): Input Records      = 10.
text(example.txt,mode=r): Formatted Records  = 10.
```

2.4.2 Print the contents of the example binary file.

```
cmlprint --objtypes=example.objtypes
      "binary(example.rdw,mode=rb,recfm=v)"
```

This example prints the variable length record file `example.rdw` using the structure output format to the screen (stdout). The output from this command should be identical to the print output above for `example.txt`.

See example 3.4.1 on page 14 for information on how to generate the binary file.

2.4.3 Unpack a file to a CSV file.

This example unpacks the file `example.rdw` into the file `example.csv` as a comma separated values file.

See example 5.4.1 on page 31 for an example of how the CSV file can be converted back (or packed) to a binary file again.

```
cmlprint --format=csvfile
       --objtypes=example.objtypes
       --output-file=example.csv
       "binary(example.rdw,mode=rb,recfm=v)"

Code Magus Limited Filetools V3.0: build 2017-03-02-16.56.45
[cmlprint] $Id: filetools_cmlprint_eg3.tex,v 1.2 2017/03/06 17:50:37 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].

Seq = 6, Length = 17
File = binary(example.rdw,mode=rb,recfm=v)
Last error:
Unable to determine type of buffer:

binary(example.rdw,mode=rb,recfm=v): Input Records      = 10.
binary(example.rdw,mode=rb,recfm=v): Formatted Records  = 10.
```

Note that one record caused an error, as it is not a known type, and is output to the CSV file as a hex dump (see below). As this can invalidate the structure of the CSV file, make sure that only valid records are printed or unpacked by adding a select clause to the command that selects valid records. In this case use the option:

```
--select="from EXAMPLE_RECORDS_VALID;"
...or...
--select="from EXAMPLE_RECORDS_CONTROL; from EXAMPLE_RECORDS_DATA;"
```

to only copy records that have a record type of H,T,1,2 or 3.

The CSV file produced is shown below:

```
~~OBJTYPES,example.objtypes
~~COMMENT,Start of File = binary(example.rdw,mode=rb,recfm=v).
~~COMMENT,Using object types = example.objtypes.
~~OBJTYPE,EXAMPLE_RECORDS_HEADER
"RECORD_TYPE","EX_KEY","EX_DATE"
"H",0000,20130101
~~OBJTYPE,EXAMPLE_RECORDS_TYPE3
"RECORD_TYPE","EX_KEY","EX_CHARACTER","EX_COUNT","EX_VC(1)","EX_VC(2)","EX_VC(3)","EX_VC(4)","EX_VC(5)"
"3",1111,"abcde",05,"a","b","c","d","e"
~~OBJTYPE,EXAMPLE_RECORDS_TYPE1
"RECORD_TYPE","EX_KEY","EX_CHARACTER","EX_COUNT","EX_VC(1)"
"1",1122,"f    ",01,"f"
"1",1133,"kl  z",05,"k","l"," "," ","z"
~~OBJTYPE,EXAMPLE_RECORDS_TYPE3
"RECORD_TYPE","EX_KEY","EX_CHARACTER","EX_COUNT","EX_VC(1)","EX_VC(2)","EX_VC(3)","EX_VC(4)","EX_VC(5)"
"3",1144,"pqrst",05,"p","q","r","s","t"
        00..__..__05..__..__10..__..__15..__..__20..__..__25..__..__30..
  0000: 353131353561626364783035616263646 5
  0000: 5.1.1.5.5.a/b.c.d.x.0.5.a/b.c.d.e.

~~OBJTYPE,EXAMPLE_RECORDS_TYPE1
"RECORD_TYPE","EX_KEY","EX_CHARACTER","EX_COUNT","EX_VC(1)","EX_VC(2)","EX_VC(3)"
"1",1166,"uvw  ",03,"u","v","w"
~~OBJTYPE,EXAMPLE_RECORDS_TYPE2
"RECORD_TYPE","EX_KEY","EX_CHARACTER","EX_COUNT","EX_VC(1)","EX_VC(2)","EX_VC(3)","EX_VC(4)"
"2",1177,"zabc ",04,"z","a","b","c"
~~OBJTYPE,EXAMPLE_RECORDS_TYPE3
"RECORD_TYPE","EX_KEY","EX_CHARACTER","EX_COUNT","EX_VC(1)","EX_VC(2)","EX_VC(3)","EX_VC(4)","EX_VC(5)"
"3",1188,"abcde",05,"e","f","g","h","i"
~~OBJTYPE,EXAMPLE_RECORDS_TRAILER
"RECORD_TYPE","EX_KEY","EX_RECORDS"
"T",9999,000008
```

3 Code Magus Copy cmlcopy

3.1 Introduction

The cmlcopy tool reads an input file mapped by an objtypes [13] definition and copies it to the output file.

3.2 Processing

The data can be filtered by the object types definition and various of the command line parameters and may be encoded in a different character set to that of the running system.

3.3 Command line parameters

```
Usage: cmlcopy [OPTION...]
  -i, --input-spec=<access>(<object>[,<options>])      Input stream open specs
                                                       string
  -o, --output-spec=<access>(<object>[,<options>])     Output stream open spec
                                                       string
  -t, --objtypes={|<objtypes-name>}                    Name of objtypes member
                                                       for buffer typing
  -w, --select={|<from-where>}                         Select by type and
                                                       optional expression
                                                       over type
  -s, --skip-input-records={0|<count>}                 Only start processing
                                                       records after skipping
                                                       input records
  -n, --max-input-records={0|<count>}                  Limit the number of
                                                       records read from input
                                                       file (after skip)
  -m, --max-output-records={0|<count>}                 Limit the number of
                                                       records to copy to
                                                       output file (after skip)
  -0, --skip-zero-length-records                       Skip input records that
                                                       have a zero length
  -v, --verbose                                        Verbose processing mode

Help options:
  -?, --help                                           Show this help message
  --usage                                              Display brief usage
                                                       message
```

where:

- `-i, --input-spec=<access>(<object>[,<options>])`
  Input stream open specs string
- `-o, --output-spec=<access>(<object>[,<options>])`
  Output stream open spec string
- `-t, --objtypes={|<objtypes-name>}`
  Name of objtypes member for buffer typing

- -w, --select={|<from-where>}
  Select by type and optional expression over type
- -s, --skip-input-records={0|<count>}
  Only start processing records after skipping input records
- -n, --max-input-records={0|<count>}
  Limit the number of records read from input file (after skip)
- -m, --max-output-records={0|<count>}
  Limit the number of records to copy to output file (after skip)
- -0, --skip-zero-length-records
  Skip input records that have a zero length If a the current record read has a zero length then skip it (do not attempt to write the record to the output) and continue processing with the next record. This situation can occur when using editam[9] and the edit process encounters an error during record editing (usually expansion of a compressed record on read) and is configured to return a zero length record.
- -v, --verbose
  Produce verbose run time messages during processing.

3.4 Examples

3.4.1 Copy a text file to a binary format file.

Use the following to convert a text file into a binary file. This example can be used to generate the example.rdw file from example.txt, which can be created with any editor from the data shown in appendix A.3 on page 42.

```
cmlcopy --input-spec="text(example.txt,mode=r)"
        --output-spec="binary(example.rdw,mode=wb,recfm=v)"

Code Magus Limited Filetools V3.0: build 2017-03-02-16.56.45
[cmlcopy] $Id: filetools_cmlcopy_eg1.tex,v 1.2 2017/03/06 17:50:37 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
text(example.txt,mode=r): Input Records = 10.
binary(example.rdw,mode=wb,recfm=v): Output Records = 10.
```

3.4.2 Copy one binary file to another.

```
cmlcopy --input-spec="binary(example.rdw,mode=rb,recfm=v)"
        --output-spec="binary(example_2.rdw,mode=wb,recfm=v)"

Code Magus Limited Filetools V3.0: build 2017-03-02-16.56.45
[cmlcopy] $Id: filetools_cmlcopy_eg2.tex,v 1.2 2017/03/06 17:50:37 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
binary(example.rdw,mode=rb,recfm=v): Input Records = 10.
```

```
binary(example_2.rdw,mode=wb,recfm=v): Output Records = 10.
```

This example copies the file `example.rdw` to `example_2.rdw`. Note that the record lengths of each record are embedded in the stream file just before each data record. If the records in the file were all the same length then the record length would not have to prefix every record in the file but the Recio open string would have to specify something like '`recfm=f,reclen=30`'; in which case the records would all be 30 bytes long.

3.4.3 Copy a text file to another using record selection.

```
cmlcopy --input-spec="text(example.txt,mode=r)"
        --output-spec="standard(out)"
        --objtypes=example.objtypes
        --select="from EXAMPLE_RECORDS_DATA where
                    ex_rec.ex_data.ex_count=5;"

Code Magus Limited Filetools V3.0: build 2017-03-02-16.56.45
[cmlcopy] $Id: filetools_cmlcopy_eg3.tex,v 1.2 2017/03/06 17:50:37 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
Query string parsed: from EXAMPLE_RECORDS_DATA where ex_rec.ex_data.ex_count=5;

31111abcde05abcde
11133kl  z05kl  z
31144pqrst05pqrst
31188abcde05efghi
text(example.txt,mode=r): Input Records = 10.
standard(out): Output Records = 4.
```

This example uses a query string and `objtypes` to select a subset of the records. The copy book and objtypes config in appendix A.1 on page 40 and appendix A.2 on page 40 respectively are examples of meta data and define how the data in the file is mapped. The output was printed to the terminal by the `standard` access method.

3.4.4 Copy one binary file to another on MVS.

MVS is a record based platform as opposed to Linux, Unix and Windows which are stream orientated platforms. As such more Recio options are required.

```
cmlcopy
   --objtypes=example.objtypes
   --input-spec="binary(DD:IN,
                  mode=[rb,type=record],recfm=v,type=record)"
   --output-spec="binary(DD:OUT,
                  mode=[wb,type=record],recfm=v,type=record)"
```

This example copies the file specified by the DDNAME `IN` to the file specified by the DDNAME `OUT`. The input file must have an MVS record format of 'V' or 'VB'. See appendix B on page 45 for an example of using JCL to run filetools tools.

3.4.5 Copy one binary file to another on MVS using dynamic allocation.

Instead of using a static DDNAME in MVS JCL the Code Magus Recio MVS Access Method [4] can be used. This access method uses JCL syntax for allocating datasets and can refer to an underlying access method with the 'with' and 'using' options.

```
cmlcopy
   --objtypes=example.objtypes
   --input-spec="mvs([DSN=HLQ.INFILE,DISP=SHR],
                  using=binary,
                  with=[mode=[rb,type=record],recfm=v,type=record])"
   --output-spec="mvs([DSN=HLQ.OUTFILE,DISP=(NEW,CATLG,CATLG),RECFM=VB,
                  LRECL=4096,DSORG=PS,SPACE=(CYL,(2,2))],
                  using=binary,
                  with=[mode=[wb,type=record],recfm=v,type=record])"
```

This example copies the file HLQ.INFILE to the file HLQ.OUTFILE allocating the output file at the same time.

3.4.6 Remotely copy an MVS binary file using dynamic allocation.

The previous request can also be performed across TCP/IP as a way of reading remote data. The Code Magus Recio Remote Access Method [5] takes another complete access method open string as its object as follows.

```
cmlcopy
  --input-spec="remote(
     [mvs([DSN=HLQ.EXAMPLE.VBFILE,DISP=SHR],
       using=binary,
       with=[mode=[rb,type=record],recfm=v,type=record])],
       host=mvsHost,user=mvsUser,password=mvsPwd)"
  --output-spec="remote(
     [mvs([DSN=HLQ.EXAMPLE.VBFILE2,
       DISP=(NEW,CATLG,CATLG),RECFM=VB,LRECL=27990,
       DSORG=PS,SPACE=(CYL,(1,0))],
       using=binary,
       with=[mode=[wb,type=record],recfm=v,type=record])],
       host=mvsHost,user=mvsUser,password=mvsPwd)"

Code Magus Limited Filetools V2.0: build 2013-05-31-08.36.34
[cmlcopy] $Id: filetools_cmlcopy_eg7.tex,v 1.1 2013/12/20 18:45:26 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2011 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
remote([mvs([DSN=HLQ ....): Input Records = 10.
remote([mvs([DSN=HLQ ....): Output Records = 10.
```

Effectively this would copy HLQ.EXAMPLE.VBFILE to the file HLQ.EXAMPLE.VBFILE2 allocating it at the same time. As cmlcopy is run from a client workstation this command could be used to copy a file from one z/OS system to another.

3.4.7 Copy one local text file to another.

```
cmlcopy --input-spec="text(example.txt,mode=r)"
        --output-spec="text(example_2.txt,mode=w)"
```

```
Code Magus Limited Filetools V3.0: build 2017-03-02-16.56.45
[cmlcopy] $Id: filetools_cmlcopy_eg8.tex,v 1.2 2017/03/06 17:50:37 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
text(example.txt,mode=r): Input Records = 10.
text(example_2.txt,mode=w): Output Records = 10.
```

This example copies the file `example.txt` to `example_2.txt`. Here the file must be a valid text file for the platform it is currently on and each record may be any length and is delimited by the line feed character sequence for that platform.

3.4.8 Copy one text file to another and change its text type.

Windows and Linux/Unix have a different encoding for the end of a line in the physical text files on disk. Normally Unix uses `0x0a` and Windows `0x0d0a`. In order to convert from one to the other the following can be specified.

```
cmlcopy --input-spec="text(example.txt,mode=r,texttype=UNIX)"
        --output-spec="text(example_2D.txt,mode=w,texttype=DOS)"
```

```
Code Magus Limited Filetools V3.0: build 2017-03-02-16.56.45
[cmlcopy] $Id: filetools_cmlcopy_eg9.tex,v 1.2 2017/03/06 17:50:37 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
text(example.txt,mode=r,texttype=UNIX): Input Records = 10.
text(example_2D.txt,mode=w,texttype=DOS): Output Records = 10.
```

This would convert a UNIX formatted text file to a DOS formatted one. A hex dump of each file follows:

```
xxd example.txt
```

```
0000000: 4830 3030 3032 3031 3330 3130 310a 3331  H000020130101.31
0000010: 3131 3161 6263 6465 3035 6162 6364 650a  111abcde05abcde.
0000020: 3131 3132 3266 2020 2020 3031 660a 3131  11122f    01f.11
0000030: 3133 336b 6c20 207a 3035 6b6c 2020 7a0a  133kl  z05kl  z.
0000040: 3331 3134 3470 7172 7374 3035 7071 7273  31144pqrst05pqrs
0000050: 740a 3531 3135 3561 6263 6478 3035 6162  t.51155abcdx05ab
0000060: 6364 650a 3131 3136 3675 7677 2020 3033  cde.11166uvw  03
0000070: 7576 770a 3231 3137 377a 6162 6320 3034  uvw.21177zabc 04
0000080: 7a61 6263 0a33 3131 3838 6162 6364 6530  zabc.31188abcde0
0000090: 3565 6667 6869 0a54 3939 3939 3030 3030  5efghi.T99990000
00000a0: 3038 0a                                  08.
```

```
xxd example_2D.txt
```

```
0000000: 4830 3030 3032 3031 3330 3130 310d 0a33  H000020130101..3
0000010: 3131 3161 6263 6465 3035 6162 6364 6465  1111abcde05abcde
0000020: 0d0a 3131 3132 3266 2020 2020 3031 660d  ..11122f    01f.
0000030: 0a31 3131 3333 6b6c 2020 7a30 356b 6c20  .11133kl  z05kl
0000040: 207a 0d0a 3331 3134 3470 7172 7374 3035   z..31144pqrst05
0000050: 7071 7273 740d 0a35 3131 3535 6162 6364  pqrst..51155abcd
0000060: 7830 3561 6263 6465 0d0a 3131 3136 3675  x05abcde..11166u
0000070: 7677 2020 3033 7576 770d 0a32 3131 3737  vw  03uvw..21177
0000080: 7a61 6263 2030 347a 6162 630d 0a33 3131  zabc 04zabc..311
0000090: 3838 6162 6364 6530 3565 6667 6869 0d0a  88abcde05efghi..
00000a0: 5439 3939 3930 3030 3030 380d 0a         T9999000008..
```

4 Code Magus Compare cmlcomp

4.1 Introduction

The cmlcomp tool reads two input files mapped by an objtypes [13] definition and compares them.

4.2 Processing

The data can be filtered by the object types definition and various of the command line parameters and may be encoded in a different character set to that of the running system. Sections of a record may also be compared.

4.3 Command line parameters

```
Usage: cmlcomp <access>(<object-1>[,<options>]) <access>(<object-2>[,<options>])
  -v, --verbose                                  List components being parsed
  -t, --objtypes=STRING                          Object types member name
  -k, --key-fields=type+field-1[:field-2 ...]    List names of fields in the
                                                 key
  -x, --max-diffs={|<max-diffs>}                 Max differences to report
                                                 before stopping
  -y, --max-diffs-by-type={200|<max-type>}       Max differences by type to
                                                 report before skip
  -z, --max-print-by-type={0|<max-type-print>}   Max differences by type to
                                                 print, but still report on all
  -f, --max-fields={10000|<max-fields>}          Max fields per record to
                                                 format
  -o, --compare-offset={0|<offset>}              Offset to start comparing
                                                 records
  -g, --compare-length={0|<length>}              Length of compares of record
                                                 data
  -r, --relative-records                         Record keys are their
                                                 position in the file
  -u, --input-is-unsorted                        Indicate that input files need
                                                 to be sorted internally on
                                                 the given key
  -b, --format={structure|csvfile}               Difference report format
  -s, --delimiter={,|<delimiter>}                CSV file field delimiter
                                                 character
  -c, --condense-report                          Only report on keys and
                                                 differences
  -r, --condense-insert-delete                   Only report on the keys when
                                                 a record is only in one file

Help options:
  -?, --help                                     Show this help message
  --usage                                        Display brief usage message
```

where:

- -t, --objtypes=STRING
  Object types member name

- `-k, --key-fields=type+field-1[:field-2 ...]`
  List names of fields in the key. One or more names can be specified separated by colons. Each key is made up of the object type name followed by a plus sign and the fully qualified field name. To specify more than one key separate each key specification with a colon.
- `-x, --max-diffs={|<max-diffs>}`
  Max differences to report before stopping
- `-y, --max-diffs-by-type={200|<max-type>}`
  Max differences by type to report before skip. This parameter will still perform a compare at record level and report the differences but will stop comparing at a field level if the number if differences for a particular object type exceeds the value set. The default is set at 200.
- `-z, --max-print-by-type={0|<max-type-print>}`
  Max differences by type to print, but still report on all. This parameter only suppresses the detail print of differences once the number of differences for a particular object type exceed this amount. All compares are still performed and the field level and record level summary at the end completely reflects the differences found between the two files.
- `-f, --max-fields={10000|<max-fields>}`
  Max fields per record to format
- `-o, --compare-offset={0|<offset>}`
  Offset to start comparing records
- `-g, --compare-length={0|<length>}`
  Length of compares of record data
- `-r, --relative-records`
  Record keys are their position in the file
- `-u, --input-is-unsorted`
  Indicate to the utility that both input files are not sorted in the order of the specified key and thus need to be sorted internally on that key. This option is mutually exclusive with '`-r, --relative-records`', as files that are keyed on sequence are always already sorted. This option can also only be used when at least one key is specified with '`-k, --key-fields=`'.
- `-b, --format={structure|csvfile}`
  Difference report format
- `-s, --delimiter={,|<delimiter>}`
  CSV file field delimiter character
- `-c, --condensed-report`
  Produce a condensed structure differences report

- `-c, --condense-report`
  Only report on keys and differences, do not report (print) fields that are the same. All fields in a record that is only on one file is considered different and will be reported.
- `-r, --condense-insert-delete`
  Only report on the keys when a record is only in one file. This option coupled with –condense-report will only print the keys and different fields for records on both files where the key matched but one or more non key fields differ and will only print the keys for records that appear only on one file (an insert or delete situation).
- `-v, --verbose`
  List components being parsed and produce verbose run time messages during processing.

4.4 Examples

4.4.1 Compare two mapped files.

Compare two files based on the sort key of `EX_KEY` within type `EXAMPLE_RECORDS_VALID` defined in `example.objtypes` and `EXAMPLE.cpy`. The output font has been reduced in order to format it into this document.

```
cmlcomp --objtypes=example.objtypes
        --key-fields=EXAMPLE_RECORDS_VALID+EX_REC.EX_KEY
        "text(example.txt,mode=r)" "text(example2.txt,mode=r)"
```

```
Code Magus Limited Filetools V3.0: build 2017-06-02-10.03.43
[cmlcomp] $Id: filetools_cmlcomp_eg1.tex,v 1.4 2017/06/16 06:50:03 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
Left  File Name = text(example.txt,mode=r)
Right File Name = text(example2.txt,mode=r)
Using Keys:
1: EX_REC.EX_KEY


Following key-matched records differ:

Seq = 2                                                    Seq = 2
Type = EXAMPLE_RECORDS_TYPE3                               Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3                    Title = Filetools Example Record Type 3
   01 EX_REC                                                  01 EX_REC
      03 RECORD_TYPE = "3"                                       03 RECORD_TYPE = "3"
      03 EX_KEY = 1111                                           03 EX_KEY = 1111
      03 EX_DATA                                                 03 EX_DATA
         05 EX_CHARACTER = "abcde"                                   05 EX_CHARACTER = "abcde"
         05 EX_COUNT = 05                                            05 EX_COUNT = 05
         05 EX_VC(1) = "a"                                           05 EX_VC(1) = "a"
         05 EX_VC(2) = "b"                                           05 EX_VC(2) = "b"
         05 EX_VC(3) = "c"                                           05 EX_VC(3) = "c"
         05 EX_VC(4) = "d"                                           05 EX_VC(4) = "d"
         05 EX_VC(5) = "e"                          <====>           05 EX_VC(5) = "x"



Record appears only in left hand file:

Seq = 3
Type = EXAMPLE_RECORDS_TYPE1
Title = Filetools Example Record Type 1
   01 EX_REC                                               <====>
```

```
      03 RECORD_TYPE = "1"                          <====>
      03 EX_KEY = 1122                              <====>
      03 EX_DATA                                    <====>
         05 EX_CHARACTER = "f    "                  <====>
         05 EX_COUNT = 01                           <====>
         05 EX_VC(1) = "f"                          <====>



Record appears only in left hand file:

Seq = 5
Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3
   01 EX_REC                                        <====>
      03 RECORD_TYPE = "3"                          <====>
      03 EX_KEY = 1144                              <====>
      03 EX_DATA                                    <====>
         05 EX_CHARACTER = "pqrst"                  <====>
         05 EX_COUNT = 05                           <====>
         05 EX_VC(1) = "p"                          <====>
         05 EX_VC(2) = "q"                          <====>
         05 EX_VC(3) = "r"                          <====>
         05 EX_VC(4) = "s"                          <====>
         05 EX_VC(5) = "t"                          <====>



Record appears only in right hand file:

                                                        Seq = 5
                                                        Type = EXAMPLE_RECORDS_TYPE2
                                                        Title = Filetools Example Record Type 2
                                                  <====>   01 EX_REC
                                                  <====>      03 RECORD_TYPE = "2"
                                                  <====>      03 EX_KEY = 1157
                                                  <====>      03 EX_DATA
                                                  <====>         05 EX_CHARACTER = "pqrst"
                                                  <====>         05 EX_COUNT = 05
                                                  <====>         05 EX_VC(1) = "a"
                                                  <====>         05 EX_VC(2) = "b"
                                                  <====>         05 EX_VC(3) = "c"
                                                  <====>         05 EX_VC(4) = "d"
                                                  <====>         05 EX_VC(5) = "e"



Following key-matched records differ:

Seq = 7                                                 Seq = 6
Type = EXAMPLE_RECORDS_TYPE1                            Type = EXAMPLE_RECORDS_TYPE1
Title = Filetools Example Record Type 1                 Title = Filetools Example Record Type 1
   01 EX_REC                                               01 EX_REC
      03 RECORD_TYPE = "1"                                    03 RECORD_TYPE = "1"
      03 EX_KEY = 1166                                        03 EX_KEY = 1166
      03 EX_DATA                                              03 EX_DATA
         05 EX_CHARACTER = "uvw  "                               05 EX_CHARACTER = "uvw  "
         05 EX_COUNT = 03                                        05 EX_COUNT = 03
         05 EX_VC(1) = "u"                             <====>    05 EX_VC(1) = "z"
         05 EX_VC(2) = "v"                                       05 EX_VC(2) = "v"
         05 EX_VC(3) = "w"                                       05 EX_VC(3) = "w"



Record appears only in left hand file:

Seq = 9
Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3
   01 EX_REC                                        <====>
      03 RECORD_TYPE = "3"                          <====>
      03 EX_KEY = 1188                              <====>
      03 EX_DATA                                    <====>
         05 EX_CHARACTER = "abcde"                  <====>
         05 EX_COUNT = 05                           <====>
         05 EX_VC(1) = "e"                          <====>
         05 EX_VC(2) = "f"                          <====>
         05 EX_VC(3) = "g"                          <====>
         05 EX_VC(4) = "h"                          <====>
         05 EX_VC(5) = "i"                          <====>



Record appears only in right hand file:
```

```
                                                      Seq = 8
                                                      Type = EXAMPLE_RECORDS_TYPE3
                                                      Title = Filetools Example Record Type 3
                                                <====>   01 EX_REC
                                                <====>      03 RECORD_TYPE = "3"
                                                <====>      03 EX_KEY = 1222
                                                <====>      03 EX_DATA
                                                <====>         05 EX_CHARACTER = "hijkl"
                                                <====>         05 EX_COUNT = 05
                                                <====>         05 EX_VC(1) = "e"
                                                <====>         05 EX_VC(2) = "f"
                                                <====>         05 EX_VC(3) = "g"
                                                <====>         05 EX_VC(4) = "h"
                                                <====>         05 EX_VC(5) = "i"


Following key-matched records differ:

Seq = 10                                                Seq = 9
Type = EXAMPLE_RECORDS_TRAILER                          Type = EXAMPLE_RECORDS_TRAILER
Title = Filetools Example Record Trailer                Title = Filetools Example Record Trailer
   01 EX_TAIL                                              01 EX_TAIL
      03 RECORD_TYPE = "T"                                    03 RECORD_TYPE = "T"
      03 EX_KEY = 9999                                        03 EX_KEY = 9999
      03 EX_RECORDS = 000008                            <====>   03 EX_RECORDS = 000007


Left  File Name = text(example.txt,mode=r)
Right File Name = text(example2.txt,mode=r)

Details for type EXAMPLE_RECORDS_TYPE3 (Filetools Example Record Type 3):
   Differences        = 4
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TYPE2 (Filetools Example Record Type 2):
   Differences        = 1
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TYPE1 (Filetools Example Record Type 1):
   Differences        = 2
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_DATA (Filetools Example All Data records):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TRAILER (Filetools Example Record Trailer):
   Differences        = 1
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_HEADER (Filetools Example Record Header):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_CONTROL (Filetools Example All CONTROL records):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_VALID (Filetools Example All valid records):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for unmatched/untyped records:
   Differences        = 5
   Compare Suppressed = 0
   Print Suppressed   = 0

Differences for field EX_REC.EX_DATA.EX_VC[1]
    1 out of the left file total of 10 (10.00%) and out of the right file total of 9 (11.11%)
Differences for field EX_REC.EX_DATA.EX_VC[5]
    1 out of the left file total of 10 (10.00%) and out of the right file total of 9 (11.11%)
Differences for field EX_TAIL.EX_RECORDS
    1 out of the left file total of 10 (10.00%) and out of the right file total of 9 (11.11%)

Compare finished. Number of differences = 8.
Mismatched instances skipped            = 0.
Mismatched instances not printed        = 0.
Number of records read from left  file  = 10.
Number of records only on   left  file  = 3.
Number of records read from right file  = 9.
Number of records only on   right file  = 2.
```

4.4.2 Compare `example.rdw` with a unique version of itself.

Refer to the unique example 6.4.1 on page 34 for information on how to generate the unique file `example_uniqued.rdw`.

Compare the two files based on the sort key of `EX_KEY` within type `EXAMPLE_RECORDS_VALID` defined in `example.objtypes` and `EXAMPLE.cpy`.

```
cmlcomp --objtypes=example.objtypes
   --key-fields=EXAMPLE_RECORDS_VALID+EX_REC.EX_KEY
   "binary(example.rdw,mode=rb,recfm=v)"
   "binary(example_uniqued.rdw,mode=rb,recfm=v)"

Code Magus Limited Filetools V3.0: build 2017-06-02-10.03.43
[cmlcomp] $Id: filetools_cmlcomp_eg2.tex,v 1.3 2017/06/16 06:50:03 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
Left  File Name = binary(example.rdw,mode=rb,recfm=v)
Right File Name = binary(example_uniqued.rdw,mode=rb,recfm=v)
Using Keys:
1: EX_REC.EX_KEY


Record appears only in left hand file:

Seq = 9
Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3
   01 EX_REC                                              <====>
      03 RECORD_TYPE = "3"                                 <====>
      03 EX_KEY = 1188                                    <====>
      03 EX_DATA                                          <====>
         05 EX_CHARACTER = "abcde"                         <====>
         05 EX_COUNT = 05                                  <====>
         05 EX_VC(1) = "e"                                 <====>
         05 EX_VC(2) = "f"                                 <====>
         05 EX_VC(3) = "g"                                 <====>
         05 EX_VC(4) = "h"                                 <====>
         05 EX_VC(5) = "i"                                 <====>



Left  File Name = binary(example.rdw,mode=rb,recfm=v)
Right File Name = binary(example_uniqued.rdw,mode=rb,recfm=v)

Details for type EXAMPLE_RECORDS_TYPE3 (Filetools Example Record Type 3):
   Differences        = 1
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TYPE2 (Filetools Example Record Type 2):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TYPE1 (Filetools Example Record Type 1):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_DATA (Filetools Example All Data records):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TRAILER (Filetools Example Record Trailer):
   Differences        = 0
```

```
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_HEADER (Filetools Example Record Header):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_CONTROL (Filetools Example All CONTROL records):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_VALID (Filetools Example All valid records):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for unmatched/untyped records:
   Differences        = 1
   Compare Suppressed = 0
   Print Suppressed   = 0


Compare finished. Number of differences = 1.
Mismatched instances skipped            = 0.
Mismatched instances not printed        = 0.
Number of records read from left  file  = 10.
Number of records only on   left  file  = 1.
Number of records read from right file  = 9.
Number of records only on   right file  = 0.
```

4.4.3 Compare two randomly sorted mapped files.

Compare two files based on the sort key of EX_KEY within type EXAMPLE_RECORDS_VALID defined in example.objtypes and EXAMPLE.cpy.

This example uses the file example.shuffled.txt, created by running the shuffle example 7.4.1 on page 35 to compare to example2.txt. It reports the same differences as the cmlcomp example 4.4.1 on page 20.

Note that both files are sorted.

```
cmlcomp --objtypes=example.objtypes
        --input-is-unsorted
        --key-fields=EXAMPLE_RECORDS_VALID+EX_REC.EX_KEY
        "text(example.shuffled.txt,mode=r)" "text(example2.txt,mode=r)"
```

```
Code Magus Limited Filetools V3.0: build 2017-06-02-10.03.43
[cmlcomp] $Id: filetools_cmlcomp_eg3.tex,v 1.3 2017/06/16 06:50:03 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
Code Magus Limited PPM Sort V1.0: build 2017-02-27-15.32.03
[/home/hayward/mystuff/codemagus/software/build/bin/ppmsort:14214] $Id: filetools_cmlcomp_eg3.tex,v 1.3 2017/06/16 06:50:03 hayward Exp $
Copyright (c) 2007, 2008 by Code Magus Limited. All rights reserved.
mailto:stephen@codemagus.com, http://www.codemagus.com.
Total bytes input to sort process       = 193.
Total records input to sort process     = 10.
Total bytes output from merge process   = 193.
Total records output from merge process = 10.
Code Magus Limited PPM Sort V1.0: build 2017-02-27-15.32.03
[/home/hayward/mystuff/codemagus/software/build/bin/ppmsort:14217] $Id: filetools_cmlcomp_eg3.tex,v 1.3 2017/06/16 06:50:03 hayward Exp $
Copyright (c) 2007, 2008 by Code Magus Limited. All rights reserved.
mailto:stephen@codemagus.com, http://www.codemagus.com.
Total bytes input to sort process       = 176.
Total records input to sort process     = 9.
Total bytes output from merge process   = 176.
Total records output from merge process = 9.
Left  File Name = text(example.shuffled.txt,mode=r)
Right File Name = text(example2.txt,mode=r)
```

```
Using Keys:
1: EX_REC.EX_KEY


Following key-matched records differ:

Seq = 7, KeySeq = 2                                     Seq = 2, KeySeq = 2
Type = EXAMPLE_RECORDS_TYPE3                            Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3                 Title = Filetools Example Record Type 3
   01 EX_REC                                               01 EX_REC
      03 RECORD_TYPE = "3"                                    03 RECORD_TYPE = "3"
      03 EX_KEY = 1111                                        03 EX_KEY = 1111
      03 EX_DATA                                              03 EX_DATA
         05 EX_CHARACTER = "abcde"                                05 EX_CHARACTER = "abcde"
         05 EX_COUNT = 05                                         05 EX_COUNT = 05
         05 EX_VC(1) = "a"                                        05 EX_VC(1) = "a"
         05 EX_VC(2) = "b"                                        05 EX_VC(2) = "b"
         05 EX_VC(3) = "c"                                        05 EX_VC(3) = "c"
         05 EX_VC(4) = "d"                                        05 EX_VC(4) = "d"
         05 EX_VC(5) = "e"                               <====>   05 EX_VC(5) = "x"



Record appears only in left hand file:

Seq = 4, KeySeq = 3
Type = EXAMPLE_RECORDS_TYPE1
Title = Filetools Example Record Type 1
   01 EX_REC                                            <====>
      03 RECORD_TYPE = "1"                               <====>
      03 EX_KEY = 1122                                  <====>
      03 EX_DATA                                        <====>
         05 EX_CHARACTER = "f    "                      <====>
         05 EX_COUNT = 01                               <====>
         05 EX_VC(1) = "f"                              <====>



Record appears only in left hand file:

Seq = 1, KeySeq = 5
Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3
   01 EX_REC                                            <====>
      03 RECORD_TYPE = "3"                               <====>
      03 EX_KEY = 1144                                  <====>
      03 EX_DATA                                        <====>
         05 EX_CHARACTER = "pqrst"                       <====>
         05 EX_COUNT = 05                                <====>
         05 EX_VC(1) = "p"                               <====>
         05 EX_VC(2) = "q"                               <====>
         05 EX_VC(3) = "r"                               <====>
         05 EX_VC(4) = "s"                               <====>
         05 EX_VC(5) = "t"                               <====>



Record appears only in right hand file:

                                                        Seq = 5, KeySeq = 5
                                                        Type = EXAMPLE_RECORDS_TYPE2
                                                        Title = Filetools Example Record Type 2
                                                <====>  01 EX_REC
                                                <====>     03 RECORD_TYPE = "2"
                                                <====>     03 EX_KEY = 1157
                                                <====>     03 EX_DATA
                                                <====>        05 EX_CHARACTER = "pqrst"
                                                <====>        05 EX_COUNT = 05
                                                <====>        05 EX_VC(1) = "a"
                                                <====>        05 EX_VC(2) = "b"
                                                <====>        05 EX_VC(3) = "c"
                                                <====>        05 EX_VC(4) = "d"
                                                <====>        05 EX_VC(5) = "e"



Following key-matched records differ:

Seq = 5, KeySeq = 7                                     Seq = 6, KeySeq = 6
Type = EXAMPLE_RECORDS_TYPE1                            Type = EXAMPLE_RECORDS_TYPE1
Title = Filetools Example Record Type 1                 Title = Filetools Example Record Type 1
   01 EX_REC                                               01 EX_REC
      03 RECORD_TYPE = "1"                                    03 RECORD_TYPE = "1"
      03 EX_KEY = 1166                                        03 EX_KEY = 1166
      03 EX_DATA                                              03 EX_DATA
```

```
         05 EX_CHARACTER = "uvw  "                                  05 EX_CHARACTER = "uvw  "
         05 EX_COUNT = 03                                           05 EX_COUNT = 03
         05 EX_VC(1) = "u"                          <====>          05 EX_VC(1) = "z"
         05 EX_VC(2) = "v"                                          05 EX_VC(2) = "v"
         05 EX_VC(3) = "w"                                          05 EX_VC(3) = "w"



Record appears only in left hand file:

Seq = 2, KeySeq = 9
Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3
   01 EX_REC                                          <====>
      03 RECORD_TYPE = "3"                             <====>
      03 EX_KEY = 1188                                 <====>
      03 EX_DATA                                       <====>
         05 EX_CHARACTER = "abcde"                     <====>
         05 EX_COUNT = 05                              <====>
         05 EX_VC(1) = "e"                             <====>
         05 EX_VC(2) = "f"                             <====>
         05 EX_VC(3) = "g"                             <====>
         05 EX_VC(4) = "h"                             <====>
         05 EX_VC(5) = "i"                             <====>



Record appears only in right hand file:

                                                          Seq = 8, KeySeq = 8
                                                          Type = EXAMPLE_RECORDS_TYPE3
                                                          Title = Filetools Example Record Type 3
                                                    <====>   01 EX_REC
                                                    <====>      03 RECORD_TYPE = "3"
                                                    <====>      03 EX_KEY = 1222
                                                    <====>      03 EX_DATA
                                                    <====>         05 EX_CHARACTER = "hijkl"
                                                    <====>         05 EX_COUNT = 05
                                                    <====>         05 EX_VC(1) = "e"
                                                    <====>         05 EX_VC(2) = "f"
                                                    <====>         05 EX_VC(3) = "g"
                                                    <====>         05 EX_VC(4) = "h"
                                                    <====>         05 EX_VC(5) = "i"



Following key-matched records differ:

Seq = 6, KeySeq = 10                                      Seq = 9, KeySeq = 9
Type = EXAMPLE_RECORDS_TRAILER                            Type = EXAMPLE_RECORDS_TRAILER
Title = Filetools Example Record Trailer                  Title = Filetools Example Record Trailer
   01 EX_TAIL                                                01 EX_TAIL
      03 RECORD_TYPE = "T"                                      03 RECORD_TYPE = "T"
      03 EX_KEY = 9999                                          03 EX_KEY = 9999
      03 EX_RECORDS = 000008                       <====>       03 EX_RECORDS = 000007



Left  File Name = text(example.shuffled.txt,mode=r)
Right File Name = text(example2.txt,mode=r)

Details for type EXAMPLE_RECORDS_TYPE3 (Filetools Example Record Type 3):
   Differences        = 4
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TYPE2 (Filetools Example Record Type 2):
   Differences        = 1
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TYPE1 (Filetools Example Record Type 1):
   Differences        = 2
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_DATA (Filetools Example All Data records):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TRAILER (Filetools Example Record Trailer):
   Differences        = 1
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_HEADER (Filetools Example Record Header):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_CONTROL (Filetools Example All CONTROL records):
```

```
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_VALID (Filetools Example All valid records):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for unmatched/untyped records:
   Differences        = 5
   Compare Suppressed = 0
   Print Suppressed   = 0

Differences for field EX_REC.EX_DATA.EX_VC[1]
    1 out of the left file total of 10 (10.00%) and out of the right file total of 9 (11.11%)
Differences for field EX_REC.EX_DATA.EX_VC[5]
    1 out of the left file total of 10 (10.00%) and out of the right file total of 9 (11.11%)
Differences for field EX_TAIL.EX_RECORDS
    1 out of the left file total of 10 (10.00%) and out of the right file total of 9 (11.11%)

Compare finished. Number of differences = 8.
Mismatched instances skipped            = 0.
Mismatched instances not printed        = 0.
Number of records read from left  file  = 10.
Number of records only on   left  file  = 3.
Number of records read from right file  = 9.
Number of records only on   right file  = 2.
```

4.4.4 Compare two mapped files, but produce a fully condensed report.

Compare two files based on the sort key of `EX_KEY` within type `EXAMPLE_RECORDS_VALID` defined in `example.objtypes` and `EXAMPLE.cpy`.

It reports the same differences as the `cmlcomp` example 4.4.1 on page 20 except, that only the key fields are shown and where a record is found on both files, only those fields that are different.

```
cmlcomp --objtypes=example.objtypes
        --key-fields=EXAMPLE_RECORDS_VALID+EX_REC.EX_KEY
        --condense-report
        --condense-insert-delete
        "text(example.txt,mode=r)" "text(example2.txt,mode=r)"
```

```
Code Magus Limited Filetools V3.0: build 2017-06-02-10.03.43
[cmlcomp] $Id: filetools_cmlcomp_eg4.tex,v 1.2 2017/06/16 06:50:03 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
Left  File Name = text(example.txt,mode=r)
Right File Name = text(example2.txt,mode=r)
Using Keys:
1: EX_REC.EX_KEY


Following key-matched records differ:

Seq = 2                                                    Seq = 2
Type = EXAMPLE_RECORDS_TYPE3                               Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3                    Title = Filetools Example Record Type 3
      03 EX_KEY = 1111                                           03 EX_KEY = 1111
         05 EX_VC(5) = "e"                           <====>        05 EX_VC(5) = "x"



Record appears only in left hand file:

Seq = 3
Type = EXAMPLE_RECORDS_TYPE1
Title = Filetools Example Record Type 1
      03 EX_KEY = 1122                              <====>



Record appears only in left hand file:

Seq = 5
```

```
Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3
      03 EX_KEY = 1144                                  <====>



Record appears only in right hand file:

                                                              Seq = 5
                                                              Type = EXAMPLE_RECORDS_TYPE2
                                                              Title = Filetools Example Record Type 2
                                                        <====>      03 EX_KEY = 1157



Following key-matched records differ:

Seq = 7                                                       Seq = 6
Type = EXAMPLE_RECORDS_TYPE1                                  Type = EXAMPLE_RECORDS_TYPE1
Title = Filetools Example Record Type 1                       Title = Filetools Example Record Type 1
      03 EX_KEY = 1166                                              03 EX_KEY = 1166
         05 EX_VC(1) = "u"                                 <====>         05 EX_VC(1) = "z"



Record appears only in left hand file:

Seq = 9
Type = EXAMPLE_RECORDS_TYPE3
Title = Filetools Example Record Type 3
      03 EX_KEY = 1188                                  <====>



Record appears only in right hand file:

                                                              Seq = 8
                                                              Type = EXAMPLE_RECORDS_TYPE3
                                                              Title = Filetools Example Record Type 3
                                                        <====>      03 EX_KEY = 1222



Following key-matched records differ:

Seq = 10                                                      Seq = 9
Type = EXAMPLE_RECORDS_TRAILER                                Type = EXAMPLE_RECORDS_TRAILER
Title = Filetools Example Record Trailer                      Title = Filetools Example Record Trailer
      03 EX_RECORDS = 000008                              <====>      03 EX_RECORDS = 000007



Left  File Name = text(example.txt,mode=r)
Right File Name = text(example2.txt,mode=r)

Details for type EXAMPLE_RECORDS_TYPE3 (Filetools Example Record Type 3):
   Differences        = 4
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TYPE2 (Filetools Example Record Type 2):
   Differences        = 1
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TYPE1 (Filetools Example Record Type 1):
   Differences        = 2
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_DATA (Filetools Example All Data records):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_TRAILER (Filetools Example Record Trailer):
   Differences        = 1
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_HEADER (Filetools Example Record Header):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_CONTROL (Filetools Example All CONTROL records):
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for type EXAMPLE_RECORDS_VALID (Filetools Example All valid records):
```

```
   Differences        = 0
   Compare Suppressed = 0
   Print Suppressed   = 0
Details for unmatched/untyped records:
   Differences        = 5
   Compare Suppressed = 0
   Print Suppressed   = 0

Differences for field EX_REC.EX_DATA.EX_VC[1]
    1 out of the left file total of 10 (10.00%) and out of the right file total of 9 (11.11%)
Differences for field EX_REC.EX_DATA.EX_VC[5]
    1 out of the left file total of 10 (10.00%) and out of the right file total of 9 (11.11%)
Differences for field EX_TAIL.EX_RECORDS
    1 out of the left file total of 10 (10.00%) and out of the right file total of 9 (11.11%)

Compare finished. Number of differences = 8.
Mismatched instances skipped            = 0.
Mismatched instances not printed        = 0.
Number of records read from left  file  = 10.
Number of records only on   left  file  = 3.
Number of records read from right file  = 9.
Number of records only on   right file  = 2.
```

5 Code Magus Pack cmlpack

5.1 Introduction

The cmlpack tool reads a comma separated (CSV) file of data and writes an output file mapped by an objtypes [13] definition.

5.2 Processing

The output data file may be encoded in a different character set to that of the running system.

5.3 Command line parameters

```
Usage: cmlpack [OPTION...]
  -t, --object-types-file=<objtypes>                Object type collection
                                                    name config file
  -m, --object-type-entry=<objtype>                 Object type name
  -o, --out-file=<access>(<object>[,<options>])     Output stream open spec
                                                    string
  -d, --csv-file=<CSV file>                         Name of the CSV file
                                                    containing the data
  -s, --delimiter={,|<delimiter>}                   CSV file field delimiter
                                                    character
  -c, --charset={ascii|ebcdic}                      Default character set for
                                                    character data
  -e, --endian={big|little}                         Default binary data endian
  -i, --init-image={0x00|0xCC|<init-char>}          Buffer initialisation
                                                    image character
  -l, --override-record-length={0|<record-length>}  Override output record
                                                    length
  -v, --verbose                                     Verbose printing during
                                                    processing

Help options:
  -?, --help                                        Show this help message
  --usage                                           Display brief usage message
```

where:

- -t, --object-types-file=<objtypes>
  Object type collection name config file
- -m, --object-type-entry=<objtype>
  Object type name
- -o, --out-file=<access>(<object>[,<options>])
  Output stream open spec string
- -d, --csv-file=<CSV file>
  Name of the CSV file containing the data

- `-s, --delimiter={,|<delimiter>}`
  CSV file field delimiter character
- `-c, --charset={ascii|ebcdic}`
  Default character set for character data
- `-e, --endian={big|little}`
  Default binary data endian
- `-i, --init-image={0x00|0xCC|<init-char>}`
  Buffer initialisation image character
- `-l, --override-record-length={0|<record-length>}`
  Override the output record length and make it a fixed length
- `-v, --verbose`
  Produce verbose run time messages during processing.

5.4 Examples

5.4.1 Convert a CSV file back to a binary data file.

```
cmlpack
   --object-types-file=example.objtypes
   --csv-file=example.csv
   --out-file="binary(example_packed.rdw,mode=wb,recfm=v)"

Code Magus Limited Filetools V3.0: build 2017-03-02-16.56.45
[cmlpack] $Id: filetools_cmlpack_eg1.tex,v 1.2 2017/03/06 17:50:37 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
```

This example will convert the CSV file back to a data file. If the CSV file was originally created via `cmlprint` from `example.rdw` as shown earlier in example 2.4.3 on page 11 then the file `example.packed` should be exactly the same as the original and `cmlcomp` would not highlight any differences.

6 Code Magus Unique cmluniq

6.1 Introduction

The cmluniq tool does simple domain frequency analyses and performs a type of downsizing based on making certain fields, or combinations of fields unique in an output file by filtering the records from the given input file. The tool works by supplying a copybook and a record in that copybook which is assumed to map the file. There can be any number of fields or groups of fields. Ideally a field or group of fields would be chosen because the cardinality of the column or column group is relatively small and the field or group of fields forms some sort of path or conditional indicator to the processing programs or system.

6.2 Processing

The input file is filtered to produce an output file by considering a number of fields or groups of fields. A record from the input file passes the filtering and is written to the output file if one of the fields or groups of fields within the file contains a value which has not been seen before. When considering a group fields, the group is assumed to make up a virtual field to which the same filtering rule applies.

6.3 Command line parameters

The general format of running the program is:

```
unique [options] <column-group> ...
```

The options are:

```
Usage: cmluniq <column-group> ...
        <column-group> = <field-name>[:<field-name> ...]
  -i, --input-spec=<access>(<object>[,<options>])      Input stream open specs
                                                       string
  -o, --output-spec=<access>(<object>[,<options>])     Output stream open
                                                       specs string
  -m, --map-file=<map file>                            Name of file which
                                                       contains the layout map
  -n, --map-name=<map name>                            Name of the structure
                                                       in the map file to use
  -c, --charset={ascii|ebcdic}                         Default character set
                                                       for character data
  -e, --endian={big|little}                            Default binary data
                                                       endian
  -h, --hash-mod={default|<integer>}                   Hash table modulo value
  -d, --list-domains                                   List the values of the
                                                       column domains
  -s, --domains-report-csv                             Format domain report as
                                                       CSV (spreadsheet)
  -k, --key-count={1|<integer>}                        Maximum records per key
                                                       group value
```

```
  -x, --max-records={0|<integer>}          Max records in output
                                           file. 0 for no limit.
  -u, --allow-duplicate-names              Override to allow
                                           duplicate field names
                                           in copybook
  -v, --verbose                            Verbose printing during
                                           processing

Help options:
  -?, --help                               Show this help message
  --usage                                  Display brief usage
                                           message
```

where:

- `-i, --input-spec=<access>(<object>[,<options>])`
  Input stream open specs string. This is a formatted `Recio` open string naming the access method, the file (object) and any parameters to the access method.
- `-o, --output-spec=<access>(<object>[,<options>])`
  Output stream open specs string. This is a formatted `Recio` open string naming the access method, the file (object) and any parameters to the access method.
- `-m, --map-file=<map file>`
  Name of file which contains the layout map. This is a `COBOL` copy book relating to the input file.
- `-n, --map-name=<mape name>`
  Name of the structure in the map file to use. The structure in the map-file metadata that should be mapped to the input data.
- `-c, --charset={ascii|ebcdic}`
  Default character set for character data.
- `-e, --endian={big|little}`
  Default binary data endian.
- `-h, --hash-mod={default|<integer>}`
  Hash table modulo value.
- `-d, --list-domains`
  List the values of the column domains.
- `-s, --domains-report-csv`
  Format the domain report as a comma separated variable (CSV) file or spreadsheet.
- `-k, --key-count={1|<integer>}`
  Maximum records per key group value.
- `-x, --max-records={0|<integer>}`
  Max records in output file. Zero infers no limit.

- `-u, --allow-duplicate-names`
  Override that allows duplicate field names in copybook. All fields used as unique keys need to be fully qualified from the '01' level.
- `-v, --verbose`
  Produce verbose run time messages during processing.
- `<column-group> ::= <field-name>[:<field-name> ...]`
  A column group is either a single field name or a list of field names separated with the colon character (without spaces). The singleton fields or the groups of fields themselves are separated from each other as separate command line parameters with at least a single space between the groups.

6.4 Examples

6.4.1 Drop non-unique records from a file.

```
cmluniq --map-file=EXAMPLE.cpy
        --map-name=EX-REC
        --input-spec="binary(example.rdw,mode=rb,recfm=v)"
        --output-spec="binary(example_uniqued.rdw,mode=wb,recfm=v)"
        EX-CHARACTER

Code Magus Limited Filetools V3.0: build 2017-03-02-16.56.45
[cmluniq] $Id: filetools_cmluniq_eg1.tex,v 1.3 2017/03/06 17:50:37 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
```

This example would drop any duplicate records found in the input file based on the field `EXAMPLE_CHARACTER` in the `COBOL` copy book structure `EX-REC` in the copy book `EXAMPLE.cpy`.

The file `example.rdw` has two identical records. See example 4.4.2 on page 23 for more information on comparing the output from this example with the original file.

7 Code Magus Shuffle `cmlshufl`

7.1 Introduction

The `cmlshufl` tool will read in a file described by a `Recio` open string specification and write out the file described by the output `Recio` open string specification by shuffling the sequence of the records from the input file in a random manner. This means that although the output file is the same size and holds the same records as the input file repeated runs using the same input file will not produce the same sequence of records in the output file from one run to the next.

7.2 Processing

The input file is read and the order of the records are randomly shuffled before writing all the records to the output file.

7.3 Command line parameters

The general format of running the program is:

```
cmlshufl [options] ...
```

The options are:

```
Usage: cmlshufl [OPTION...]
  -i, --input-spec=<access>(<object>[,<options>])      Input stream open specs
                                                       string
  -o, --output-spec=<access>(<object>[,<options>])     Output stream open
                                                       specs string
  -v, --verbose                                        Verbose printing during
                                                       processing

Help options:
  -?, --help                                           Show this help message
  --usage                                              Display brief usage
```

where:

- `-i, --input-spec=<access>(<object>[,<options>])`
  Input stream open specs string. This is a formatted `Recio` open string naming the access method, the file (object) and any parameters to the access method.
- `-o, --output-spec=<access>(<object>[,<options>])`
  Output stream open specs string. This is a formatted `Recio` open string naming the access method, the file (object) and any parameters to the access method.
- `-v, --verbose`
  Produce verbose run time messages during processing.

7.4 Examples

7.4.1 Shuffle the sequence of records from a file.

```
cmlshufl
   --input-spec="text(example.txt,mode=r)"
   --output-spec="text(example.shuffled.txt,mode=w)"
```

```
Code Magus Limited Filetools V3.0: build 2017-03-02-16.56.45
[cmlshufl] $Id: filetools_cmlshufl_eg1.tex,v 1.3 2017/03/06 17:50:37 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
Code Magus Limited PPM Sort V1.0: build 2017-02-27-15.32.03
[/home/hayward/mystuff/codemagus/software/build/bin/ppmsort:13861] $Id: filetools_cmlshufl_eg1.tex,v 1.3 2
```

```
Copyright (c) 2007, 2008 by Code Magus Limited. All rights reserved.
mailto:stephen@codemagus.com, http://www.codemagus.com.
Total bytes input to sort process       = 233.
Total records input to sort process     = 10.
Total bytes output from merge process   = 233.
Total records output from merge process = 10.
```

This example would write out the file `example.shuffled.txt` based on the input file `example.txt` by randomly shuffling the record sequence.

8 Code Magus Data Masking `cmlmask`

8.1 Introduction

The `cmlmask` tool reads in a file described by a `Recio` open string specification and, under the object types definition, allow the user to obfuscate any numeric field with a specified value or any character field, determined by offset and length, with a specified character. It then writes out the masked record to the file described by the output `Recio` open string specification

8.2 Processing

The output file is an exact copy of the input file or a subset of it if any limiting parameters are specified, but with specified portions (fields) obfuscated with arbitrary constants. The default constant for numeric fields is 0 and for character fields, the hash character '#'.

8.3 Command line parameters

The general format of running the program is:

```
cmlmask [OPTIONS]* <typename>:<fieldname>
        [{:<numericvalue>|[:<offset>[:<length>[:{<maskbyte>|<maskhex>}]]]}] ...
```

The options are:

```
Usage: cmlmask [OPTIONS]* <typename>:<fieldname>
        [{:<numericvalue>|[:<offset>[:<length>[:{<maskbyte>|<maskhex>}]]]}] ...

  -i, --input-spec=<access>(<object>[,<options>])      Input stream open
                                                       specs string
  -o, --output-spec=<access>(<object>[,<options>])     Output stream open
                                                       spec string
  -t, --objtypes={|<objtypes-name>}                    Name of objtypes member
                                                       for buffer typing
  -w, --select={|<from-where>}                         Select by type and
                                                       optional expression
                                                       over type
```

```
  -s, --skip-input-records={0|<count>}          Only start processing
                                                records after skipping
                                                input records
  -n, --max-input-records={0|<count>}           Limit the number of
                                                records read from input
                                                file (after skip)
  -m, --max-output-records={0|<count>}          Limit the number of
                                                records to copy to
                                                output file (after skip)
  -0, --skip-zero-length-records                Skip input records
                                                that have a zero length
  -v, --verbose                                 Verbose processing mode

Help options:
  -?, --help                                    Show this help message
      --usage                                   Display brief usage message
```

where:

- Field Mask specification
  The field mask specification can be repeated as many times as there are fields that are required to be masked. Either a numeric value can be set for numeric fields or a portion of a character field can be masked by a repeated character.
  - `<typename>:<fieldname>`
    Both type and field name must be supplied delimited by a colon. This identifies the field to be masked.
  - `:<numericvalue>` OR
    `:<offset>[:<length>[:{<maskbyte>|<maskhex>}]]`
    This identifies either a numeric value or a repeated character string. If neither are specified then numeric fields are set to zero and the hash character is used to overwrite character fields.

    If the field is numeric it can be set to a single value. The value must be delimited from the field name by a colon.

    If the field is a character field and then all or any portion of it may be overlaid by the mask character. The mask byte can be specified as a single printable character or a single hex character written as '0xXX', where 'XX' is any valid hexadecimal value; '0x00' through '0xFF'. If the offset and length values specify a length that exceeds the field length then the specified length is decremented to fit.

    Example are:
    - `:13` will set a numeric field in all applicable records to 13.
    - `:::%` will mask the entire field with the '%' character.
    - `:0:2:@` will mask the first two bytes of the field with the '@' character.
    - `:3::0x00` will mask the field starting from the fourth byte to the end of the field with binary zeros; the COBOL low value.

`*:2:3:0xFF` will mask the field from the third byte for three bytes with binary hex 'FF' in each byte; the COBOL high value.

- `-i, --input-spec=<access>(<object>[,<options>])`
  Input stream open specs string. This is a formatted `Recio` open string naming the access method, the file (object) and any parameters to the access method.
- `-o, --output-spec=<access>(<object>[,<options>])`
  Output stream open specs string. This is a formatted `Recio` open string naming the access method, the file (object) and any parameters to the access method.
- `-t, --objtypes={|<objtypes-name>}`
  Name of objtypes member for buffer typing
- `-w, --select={|<from-where>}`
  Select by type and optional expression over type
- `-s, --skip-input-records={0|<count>}`
  Only start processing records after skipping input records
- `-n, --max-input-records={0|<count>}`
  Limit the number of records read from input file (after skip)
- `-m, --max-output-records={0|<count>}`
  Limit the number of records to copy to output file (after skip)
- `-0, --skip-zero-length-records`
  Skip input records that have a zero length
- `-v, --verbose`
  Produce verbose run time messages during processing.

8.4 Examples

8.4.1 Mask the field EX-CHARACTER.

```
cmlmask --input-spec="text(example.txt,mode=r)"
        --output-spec="standard(out)"
        --objtypes=example.objtypes
        EXAMPLE_RECORDS_DATA:EX_REC.EX_DATA.EX_CHARACTER:1:3:@
        EXAMPLE_RECORDS_HEADER:EX_HEAD.EX_DATE:19990101

Code Magus Limited Filetools V3.0: build 2017-03-02-16.56.45
[cmlmask] $Id: filetools_cmlmask_eg1.tex,v 1.2 2017/03/06 17:50:37 hayward Exp $
Copyright (c) 2001, 2002 by Stephen Donaldson. All rights reserved.
Copyright (c) 2003--2016 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
H000019990101
31111a@@@e05abcde
11122f@@@ 01f
11133k@@@z05kl  z
31144p@@@t05pqrst
51155abcdx05abcde
```

```
11166u@@@ 03uvw
21177z@@@ 04zabc
31188a@@@e05efghi
T9999000008
text(example.txt,mode=r): Input Records = 10.
standard(out): Output Records = 10.
```

This example reads the file `example.txt`, masks the middle three bytes of the field EX_CHARACTER in valid data records with the '@' character, sets the header date to '19990101' and writes all records (modified or not) to standard out (the screen).

A Meta-Data and Data for examples.

A.1 COBOL copy book

```
      * File: EXAMPLE.cpy
      * This file is the example copy book used for demonstrating
      * the different filetools utilities in the documentation.
      * It is used with example.objtypes, example.txt and example2.txt.
      *
      * $Author: hayward $
      * $Date: 2013/12/20 18:45:26 $
      * $Id: EXAMPLE.cpy,v 1.1 2013/12/20 18:45:26 hayward Exp $
      * $Name:  $
      * $Revision: 1.1 $
      * $State: Exp $
      *
      * $Log: EXAMPLE.cpy,v $
      * Revision 1.1  2013/12/20 18:45:26  hayward
      * Add all documentation to CVS.
      * It is now all in the documents sub folder.
      *
      * Revision 1.3  2013/05/20 09:36:45  hayward
      * Fix incorrect check in message for
      * previous version. Changes are for
      * the multi-types examples.
      *
      * Revision 1.2  2013/05/20 09:32:58  hayward
      * Changes for multiple types.
      *
      * Revision 1.1  2009/06/26 10:20:31  hayward
      * Add comprehensive examples to the filetools documentation.
      *
      *
      * Example Copy book for use in documentation.
        01 EX-REC.
           03 RECORD-TYPE                  PIC X.
           03 EX-KEY                       PIC 9(4).
           03 EX-DATA.
              05 EX-CHARACTER              PIC X(5).
              05 EX-COUNT                  PIC 9(2).
              05 EX-VC                     PIC X
                                              OCCURS 1 TO 10 TIMES
                                              DEPENDING ON EX-COUNT.

       01 EX-HEAD.
           03 RECORD-TYPE                  PIC X.
           03 EX-KEY                       PIC 9(4).
           03 EX-DATE                      PIC 9(8).

       01 EX-TAIL.
           03 RECORD-TYPE                  PIC X.
           03 EX-KEY                       PIC 9(4).
           03 EX-RECORDS                   PIC 9(6).
```

A.2 Object types configuration

```
-- File: example.objtypes
-- This file is the object types definition file for demonstrating
-- the filetools utilities in the documentation.
```

```
-- It is used along with example.txt, example2.txt and EXAMPLE.cpy.
--
-- $Author: hayward $
-- $Date: 2014/01/28 12:14:35 $
-- $Id: example.objtypes,v 1.2 2014/01/28 12:14:35 hayward Exp $
-- $Name:  $
-- $Revision: 1.2 $
-- $State: Exp $
--
-- $Log: example.objtypes,v $
-- Revision 1.2  2014/01/28 12:14:35  hayward
-- Use a copybook path.
--
-- Revision 1.1  2013/12/20 18:45:26  hayward
-- Add all documentation to CVS.
-- It is now all in the documents sub folder.
--
-- Revision 1.3  2013/05/20 09:36:45  hayward
-- Fix incorrect check in message for
-- previous version. Changes are for
-- the multi-types examples.
--
-- Revision 1.2  2013/05/20 09:32:58  hayward
-- Changes for multiple types.
--
-- Revision 1.1  2009/06/26 10:20:31  hayward
-- Add comprehensive examples to the filetools documentation.
--
path ${EXAMPLE_FORMATS}"%s.cpy"
;
options ascii, omit_fillers, endian_little
;

--type EXAMPLE_RECORDS_ANY title "Filetools Example All Records"
--   book EXAMPLE
--   map EX_REC
--      include EX_REC
--;

type EXAMPLE_RECORDS_VALID title "Filetools Example All valid records"
   book EXAMPLE
   map EX_REC
      exclude EX_REC
      include EX_REC.RECORD_TYPE
      include EX_REC.EX_KEY
   when ((EX_REC.RECORD_TYPE = 'H') or
         (EX_REC.RECORD_TYPE = 'T') or
         (EX_REC.RECORD_TYPE = '1') or
         (EX_REC.RECORD_TYPE = '2') or
         (EX_REC.RECORD_TYPE = '3'))
;

type EXAMPLE_RECORDS_CONTROL title "Filetools Example All CONTROL records"
   book EXAMPLE
   map EX_REC
      exclude EX_REC
      include EX_REC.RECORD_TYPE
      include EX_REC.EX_KEY
   when ((EX_REC.RECORD_TYPE = 'H') or
         (EX_REC.RECORD_TYPE = 'T'))
;

type EXAMPLE_RECORDS_HEADER title "Filetools Example Record Header"
   book EXAMPLE
   map EX_HEAD
      include EX_HEAD
```

```
   when EX_REC.RECORD_TYPE = 'H'
;

type EXAMPLE_RECORDS_TRAILER title "Filetools Example Record Trailer"
   book EXAMPLE
   map EX_TAIL
      include EX_TAIL
   when EX_REC.RECORD_TYPE = 'T'
;

type EXAMPLE_RECORDS_DATA title "Filetools Example All Data records"
   book EXAMPLE
   map EX_REC
      include EX_REC
   when ((EX_REC.RECORD_TYPE = '1') or
         (EX_REC.RECORD_TYPE = '2') or
         (EX_REC.RECORD_TYPE = '3'))
;

type EXAMPLE_RECORDS_TYPE1 title "Filetools Example Record Type 1"
   book EXAMPLE
   map EX_REC
      include EX_REC
   when EX_REC.RECORD_TYPE = '1'
;

type EXAMPLE_RECORDS_TYPE2 title "Filetools Example Record Type 2"
   book EXAMPLE
   map EX_REC
      include EX_REC
   when EX_REC.RECORD_TYPE = '2'
;

type EXAMPLE_RECORDS_TYPE3 title "Filetools Example Record Type 3"
   book EXAMPLE
   map EX_REC
      include EX_REC
   when EX_REC.RECORD_TYPE = '3'
;
```

A.3 Example Data Files

A.3.1 Example Text File

To create an example text file cut and paste the following text into a text editor and save the file locally. The binary image of the file can be generated using cmlcopy (see example 3.4.1 on page 14).

```
H000020130101
31111abcde05abcde
11122f    01f
11133kl  z05kl  z
31144pqrst05pqrst
51155abcdx05abcde
11166uvw  03uvw
21177zabc 04zabc
31188abcde05efghi
T9999000008
```

A.3.2 Example Two Text File

This file is used for demonstrating the `cmlcomp` utility.

```
H000020130101
31111abcde05abcdx
11133kl  z05kl  z
51155abcdx05abcde
21157pqrst05abcde
11166uvw  03zvw
21177zabc 04zabc
31222hijkl05efghi
T9999000007
```

A.3.3 Example Binary File

It is not possible to show the binary version of the file in plain text, but a file dump of it and the text file is shown below for comparison. However, it can be generated from the text file with the following `cmlcopy` command (see example 3.4.1 on page 14):

```
cmlcopy -i "text(example.txt,mode=r)"
        -o "binary(example.rdw,mode=wb,recfm=v)"
```

The only real difference between these two files is the way that records are delimited. In the text file the records are delimited by a new line (Unix style 0x0a in this case) at the end of the record, whereas in the binary file a record descriptor word (RDW) prefixes each record. The first two bytes of the RDW are the record length, in big endian format, and includes the length of the RDW. The second two bytes of the RDW contain spanned record information and are not used on Unix/Linux.

If the binary file only contained fixed length records then there would be no RDW in the file and the record length would have to be supplied by the user of the file in the Recio open string; for example `binary(infile,mode=rb,recfm=f,reclen=20)`.

A hex dump of the two files is shown below.

```
example.txt
0000000: 4832 3031 3330 3130 310a 3331 3131 3161  H20130101.31111a
0000010: 6263 6465 3035 6162 6364 650a 3131 3132  bcde05abcde.1112
0000020: 3266 2020 2020 3031 660a 3131 3133 336b  2f    01f.11133k
0000030: 6c20 207a 3035 6b6c 2020 7a0a 3331 3134  l  z05kl  z.3114
0000040: 3470 7172 7374 3035 7071 7273 740a 3531  4pqrst05pqrst.51
0000050: 3135 3561 6263 6478 3035 6162 6364 650a  155abcdx05abcde.
0000060: 3131 3136 3675 7677 2020 3033 7576 770a  11166uvw  03uvw.
0000070: 3231 3137 377a 6162 6320 3034 7a61 6263  21177zabc 04zabc
0000080: 0a33 3131 3838 6162 6364 6530 3565 6667  .31188abcde05efg
0000090: 6869 0a54 3030 3030 3038 0a              hi.T000008.

example.rdw
0000000: 000d 0000 4832 3031 3330 3130 3100 1500  ....H20130101...
0000010: 0033 3131 3131 6162 6364 6530 3561 6263  .31111abcde05abc
0000020: 6465 0011 0000 3131 3132 3266 2020 2020  de....11122f
0000030: 3031 6600 1500 0031 3131 3333 6b6c 2020  01f....11133kl
0000040: 7a30 356b 6c20 207a 0015 0000 3331 3134  z05kl  z....3114
0000050: 3470 7172 7374 3035 7071 7273 7400 1500  4pqrst05pqrst...
0000060: 0035 3131 3535 6162 6364 7830 3561 6263  .51155abcdx05abc
0000070: 6465 0013 0000 3131 3136 3675 7677 2020  de....11166uvw
```

```
0000080: 3033 7576 7700 1400 0032 3131 3737 7a61  03uvw....21177za
0000090: 6263 2030 347a 6162 6300 1500 0033 3131  bc 04zabc....311
00000a0: 3838 6162 6364 6530 3565 6667 6869 000b  88abcde05efghi..
00000b0: 0000 5430 3030 3030 38                   ..T000008
```

B Example JCL for running under MVS

```
//CMLUNIQ  JOB  (@@@@),'CMLUNIQ',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/*JOBPARM LINES=999999
//*
//         SET COPYBOOK='EXAMPLE'
//         SET OBJTYPES='EXAMPLE'
//         SET ENVVAR='CMLENVAR'
//         SET CMLKEYS='CMLSWKEY'
//         SET FTEHLQ='FTEHLQ'
//         SET SYSHLQ=&FTEHLQ
//*
//UNIQUE   EXEC PGM=IKJEFT01
//STEPLIB  DD  DSN=&SYSHLQ..LOADLIB,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//COPYBOOK DD  DSN=&FTEHLQ..FILETOOL.EXAMPLE.COPY(&COPYBOOK),
//             DISP=SHR
//KEYFILE  DD  DSN=&SYSHLQ..CNTL(&CMLKEYS),DISP=SHR
//STDENV   DD  DSN=&SYSHLQ..ENVIRON(&ENVVAR),DISP=SHR
//IN       DD  DSN=&FTEHLQ..FILETOOL.EXAMPLE.DATAFILE,DISP=SHR
//OUT      DD  DSN=&&UNIQFILE,DISP=(NEW,PASS),DCB=*.IN,
//             SPACE=(TRK,(1,1))
//SYSTSIN  DD  *
 cmluniq ENVAR("_CEE_ENVFILE=DD:STDENV")/ +
  --verbose +
  --map-file DD:COPYBOOK --map-name EXAMPLE-REC +
  --input-spec=+
  "binary(+
    DD:IN,mode=[rb,type=record],recfm=f,reclen=10,type=record)" +
  --output-spec=+
  "binary(+
    DD:OUT,mode=[wb,type=record],recfm=f,reclen=10,type=record)" +
  NUM-FIELD
/*
//*
// IF (UNIQUE.RC = 0) THEN
//*
//PRINTOUT EXEC PGM=IKJEFT01
//STEPLIB  DD  DSN=&SYSHLQ..LOADLIB,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
s/OBJTYPES DD  DSN=&FTEHLQ..FILETOOL.EXAMPLE.OBJTYPES(&OBJTYPES),
//             DISP=SHR
//KEYFILE  DD  DSN=&SYSHLQ..CNTL(&CMLKEYS),DISP=SHR
//STDENV   DD  DSN=&SYSHLQ..ENVIRON(&ENVVAR),DISP=SHR
//IN       DD  DSN=&&UNIQFILE,DISP=OLD
//SYSTSIN  DD  *
 cmlprint ENVAR("_CEE_ENVFILE=DD:STDENV")/ +
    --verbose +
    --objtypes DD:OBJTYPES +
    "BINARY(DD:IN,mode=[rb,type=record],recfm=f,reclen=10,type=record)"
/*
//*
// ENDIF
//*
//
```

and `&SYSHLQ..ENVIRON(&ENVVAR)` contains similar content to the following and

implies correct installation of the `Recio` environment within in an MVS `HFS` file system:

```
_EDC_ADD_ERRNO2=1
_EDC_ZERO_RECLEN=Y
CODEMAGUS_KEYFILE=DD:KEYFILE
CODEMAGUS_AMDBINS=/ftepath/bin/
CODEMAGUS_AMDPATH=/ftepath/bin/%s.amd
CODEMAGUS_AMDLIBS=/ftepath/lib/
CODEMAGUS_AMDCATPATH=/ftepath/bin/
CODEMAGUS_AMDCATNAME=MASTCAT
CODEMAGUS_AMDSUFDL=.so
```

C Built in Functions for Expressions

C.1 Expression Overview

The lexical elements of an expression are the variables, literals, operators and other character symbols used to form an expression. These lexical elements or tokens are separated by white spaces. White spaces include sequences of the space character, new-line character, the tab character and the linefeed character and their only function is to separate or delimit the tokens.

The lexical elements are often single characters having their own apparent meaning, but some are grouped together to form a word having a specific meaning. Included or associated with each token may be an attribute value.

An expression, made up of the constituent tokens into the syntax and semantics of the grammar, is then validated and evaluated by the expression evaluation library. The evaluation of an expression produces a value that can then be used within the context of the grammar of the specific Code Magus product within which it is specified.

Examples of expressions are:

1. `3+4`
2. `balance + 100`
3. `(account.balance >= 2000)`
4. `where (account.balance = 0)`
5.
```
where (account.balance < 0 ) and
      (account.overdraft_facility = 'Y')
```
6. `SysString(account.balance)`

C.2 Expression Grammar

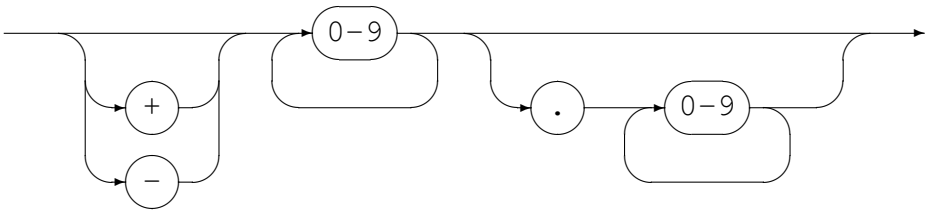
C.2.1 Lexical Elements

The base elements are *Literals* and *Identifiers*.

- Numeric Literals

  A Numeric literal is made up from an optional plus or minus sign followed by one or more digits and optionally followed by a point and one or more digits.

Number Literal



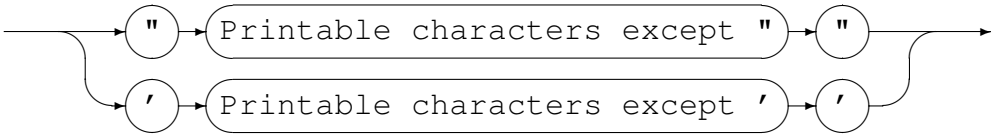
- String Literals

  String literals are made up from

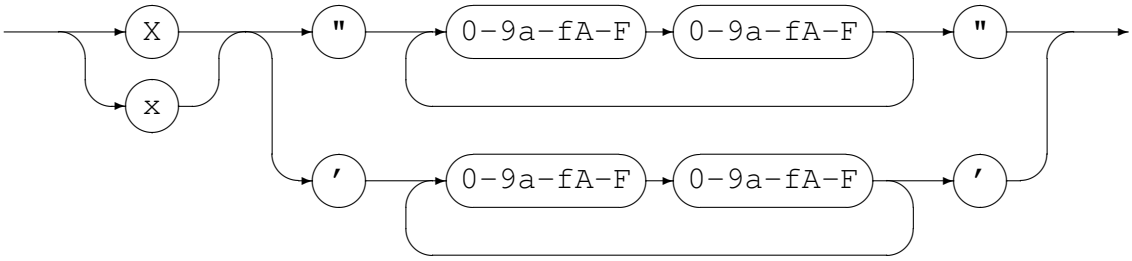
  – Any number of printable characters, except the enclosing character and a newline, enclosed in either single or double quotes.

  – An even number of hexadecimal digits enclosed in either single or double quotes and prefixed with a lower or upper case `X`.

  String Literal



  Hexadecimal Literal



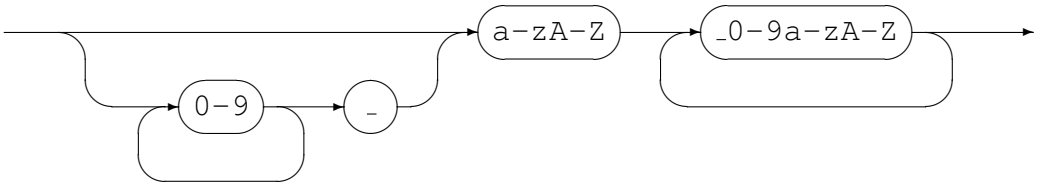
- Identifiers

  An identifier is used for both variable and function names. An identifier must conform to:

  – A lower or upper case alphabetic character followed by any number of underscores, decimal digits and upper and lower case alphabetic characters.

  – One or more decimal digits followed by an underscore and the above rule.

Identifier

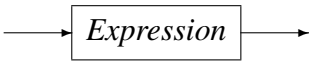


C.2.2 Syntactical Elements

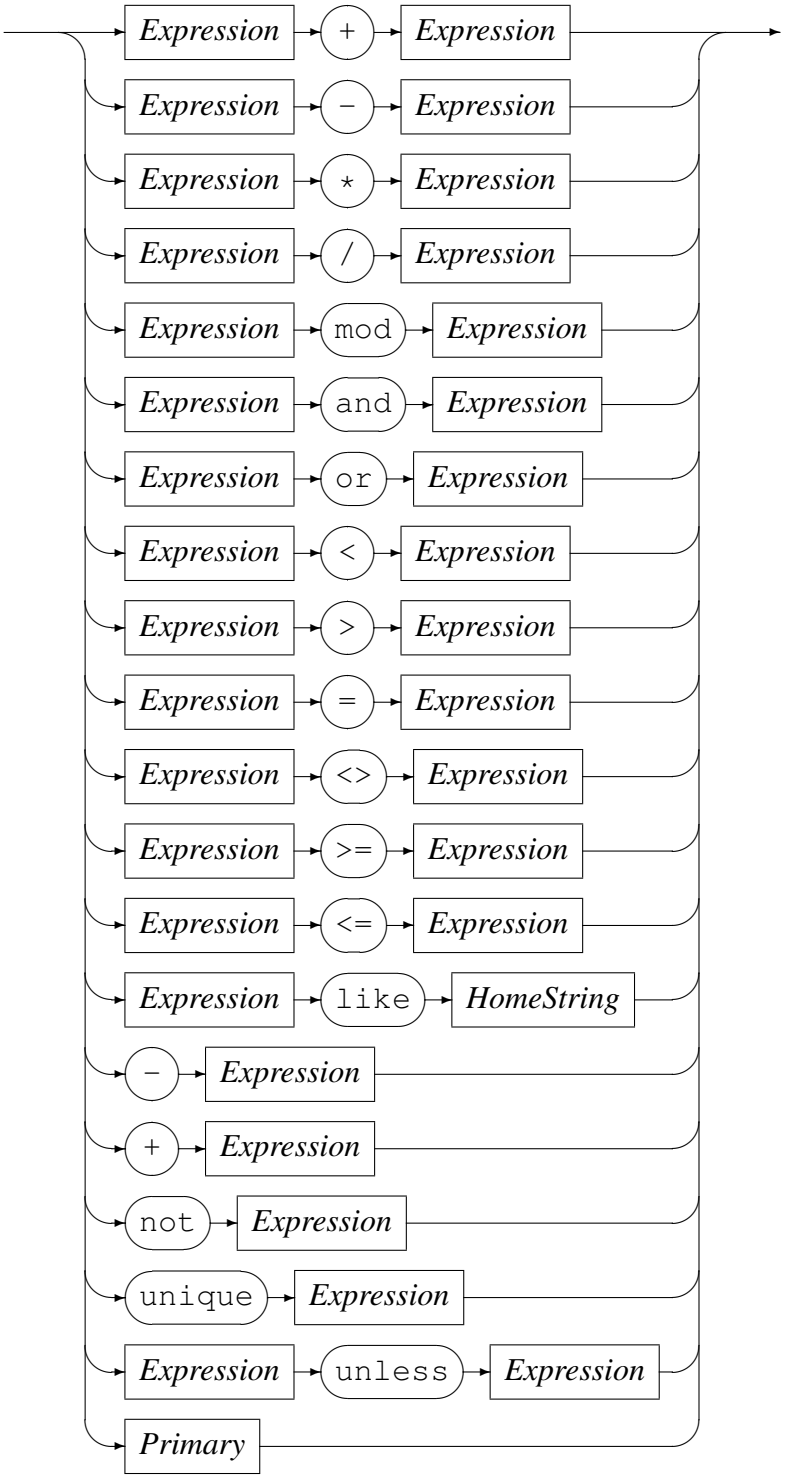
Expressions may themselves be used as syntactical elements when forming a compound expression.

The complete syntax of a compound expression is explained in the following sections starting with the compound expression and working down to the lowest level syntactic element.

CompoundExpression



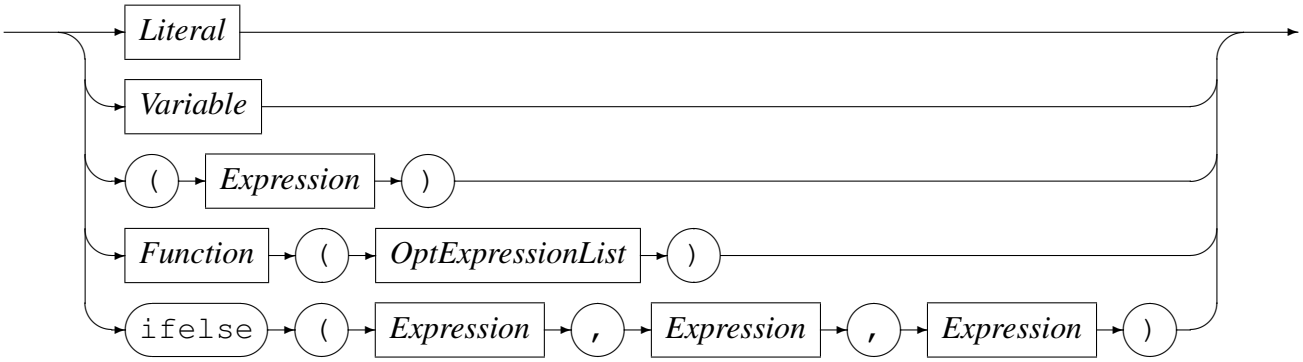
Expression



The `unless`-operator conditionally returns the value of the right-hand operand, unless there is an error evaluating the right-hand operand. In the case where the right-hand operand fails to evaluate to a proper value, the value of the left-hand operand is returned

instead. The left-hand operand is always evaluated before the right-hand operand. If the left-hand operand fails to evaluate to a proper value, then the result of the `unless`-operator is a failure.

Primary



As a terminal in the syntax structure an expression or *Primary* is either a *Literal* or a *Variable*, an *Expression* enclosed in parenthesis, a *Function* call reference, or the conditional evaluation operator `ifelse`. A *Literal* may be a *String Literal* or a *Number Literal* as described in Section C.2.1 on page 47.

Where required by the encoding indicated or defaulted, characters representing the attribute value of a string are changed to an alternate character set if the required character set is not the same as the home character set being used. For example, on a machine in which the characters are naturally represented using the EBCDIC character set encoding (such as code page of 1047 or Latin 1/Open Systems), if the data being processed is from a machine in which the characters are naturally represented using the ASCII character set (such as ISO8859-1), then the characters in the String literal (assumed to be represented in EBCDIC) will be translated to their corresponding ASCII characters for processing. This does not apply to String literals that were represented as a sequence of hexadecimal digits.

Both a *Function* (see Section C.2.2 on page 51) and an *Expression* are made up of sub-expressions, although eventually even they must terminate and resolve to a value.

A *HomeString* is a *String Literal* that may not be represented as a sequence of hexadecimal digits, but in which the encoding is left in the natural encoding of the machine processing the data; that is the machine on which the expression string is being compiled. This is required for the right-hand operand of the like operator as this operator translates the value of the left-hand operand into the local encoding when performing pattern matching.

Operators, variables and functions are described in more detail below:

- Operators

  In the context of the expression evaluation library, an operator is a symbol that

operates on or causes an action to be be performed on the constants and variables adjacent to it. An operator is either

- Monadic
  A monadic operator only operates on one value and usually employ either prefix or postfix notation in that they either occur before or after the value they operate on. The expression evaluation library uses only prefix monadic operators.
- Dyadic
  Dyadic operators operate on two values and employ infix notation in that they operate on the the values that immediately precede and follow the operator.

All operators return a value of a defined type which is the result of the computation. The type returned by an operator must be semantically consistent within the context of the rest of the expression and the grammar it may be embedded in.

Table 1 on page 50 lists the allowed operators, their precedence, associativity, arity (whether or not they are monadic or dyadic) and Type.

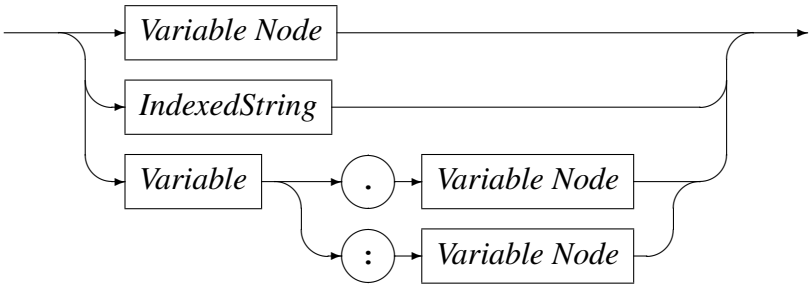
Operator	Precedence	Associativity	Arity	Type
`like`	1	non-assoc	dyadic	Relational
`<>`	1	left	dyadic	Relational
`>=`	1	left	dyadic	Relational
`<=`	1	left	dyadic	Relational
`=`	1	left	dyadic	Relational
`>`	1	left	dyadic	Relational
`<`	1	left	dyadic	Relational
`+`	2	left	dyadic	Arithmetic
`-`	2	left	dyadic	Arithmetic
`or`	2	left	dyadic	Boolean
`*`	3	left	dyadic	Arithmetic
`/`	3	left	dyadic	Arithmetic
`div`	3	left	dyadic	Arithmetic
`and`	3	left	dyadic	Boolean
`mod`	3	left	dyadic	Arithmetic
`-`	4	left	monadic	Arithmetic
`not`	4	left	monadic	Boolean
`unique`	4	left	monadic	boolean
`unless`	5	left	dyadic	boolean

Table 1: Operators: Precedence, Associativity, Arity and Type

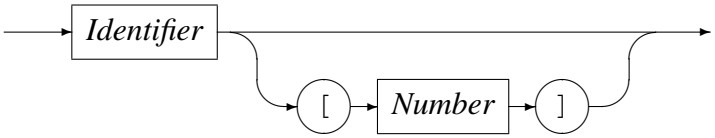
- Variables

A variable is the name of a storage location that holds a value. Simply this name is just an *Identifier*, but may be more than one level or node including an index.

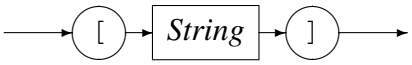
Variable



Variable Node



IndexedString



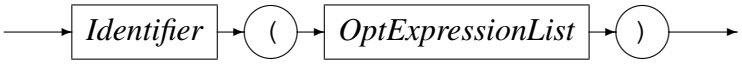
Examples of variable names are:

- `Address` - A single node variable with no indexing.
- `Customer.Address` - A two node variable.
- `Customer.Address[1]` - A two node variable where the Address portion of the variable is the first of an array of items. Here this may be the first line of an address.
- `Customer[3].Address[1]` - A two node variable that specifies the third entry of the Customer array and the first entry of the Address array within that Customer.
- `Customer.Contact.HomePhone` - A three node variable.

- Functions

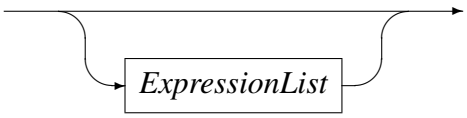
  A function is a special type of operator. It is specified by the function name, an identifier, followed by a comma separated list of arguments enclosed in parentheses.

  Function



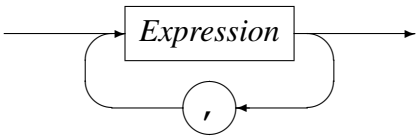
where an optional expression list is defined as

OptExpressionList



and an expression list is defined as

ExpressionList



The function call is replaced with the result of the call and the result type must be semantically consistent within the context of the rest of the expression and the grammar it may be embedded in.

C.3 Built-in Functions

Functions for expression evaluation can be supplied by the application that uses it and as such has a rich set of plug in functions that can not be documented here. However there are functions that are common to all data processing and these are supplied by the expression evaluation library and are described below.

C.3.1 SysStrLen, strlen, length

- **Synopsis**
  - `SysStrLen(string)`
  - `strlen(string)`
  - `length(string)`
- **Parameters**
  - Parameter 1 type: String.
- **Description**

  The `SysStrLne` function (aliases `strlen`, `length`) returns the number of characters in the string supplied as the first argument.

C.3.2 SysSubStr, substr

- **Synopsis**
  - `SysSubStr(string,start,length)`
  - `substr(string,start,length)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: Number.
  - Parameter 3 type: Number.
  - Return type: String.
- **Description**

  The `SysSubStr` function (alias `substr`) returns a substring of the given string from start for length characters or the remainder of string whichever is the shortest.

  The start must be greater than zero and the length must be zero or greater. If the start position is past the end of the string then a NULL string is returned.

C.3.3 SysString, string

- **Synopsis**
  - `SysString(number)`
  - `string(number)`
- **Parameters**
  - Parameter 1 type: Number.
  - Return type: String.
- **Description** The `SysString` function (alias `string`) returns the value of number as a string.

C.3.4 SysNumber, number

- **Synopsis**
  - `SysNumber(string)`
  - `number(string)`

- **Parameters**
  - Parameter 1 type: String.
  - Return type: Number.
- **Description** The `SysNumber` function (alias `number`) returns a number equivalent to the value of string.

C.3.5 SysStrCat, strcat

- **Synopsis**
  - `SysStrCat(first,second)`
  - `strcat(first,second)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Return type: String.
- **Description** The `SysStrCat` function (alias `strcat`) returns a String which is the concatenation of the two input strings `first` and `second`.

C.3.6 SysStrStr, strstr

- **Synopsis**
  - `SysStrStr(haystack,needle)`
  - `strstr(haystack,needle)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Return type: Number.
- **Description** The `SysStrStr` function (alias `strstr`) returns the start position of `needle` within `haystack`. If `needle` does not occur in `haystack` then zero is returned, otherwise the position (origin 1) is returned.

C.3.7 SysStrSpn, strspn

- **Synopsis**
  - `SysStrSpn(string,accept)`
  - `strspn(string,accept)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Return type: Number.
- **Description** The `SysStrSpn` function (alias `strspn`) returns the number of characters (bytes) in the initial segment of `string` which consist only of characters from `accept`.

C.3.8 SysStrCspn, strcspn

- **Synopsis**
  - `SysStrCspn(string,reject)`
  - `strcspn(string,reject)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Return type: Number.
- **Description** The `SysStrCspn` function (alias `strcspn`) returns the number of characters (bytes) in the initial segment of `string` which do not match any character from `reject`.

C.3.9 SysStrPadRight, padright

- **Synopsis**
  - `SysStrPadRight(string,length,pad)`
  - `padright(string,length,pad)`
- **Parameters**
  - Parameter 1 type: String.

  - Parameter 2 type: Number.
  - Parameter 3 type: String. Although the type is String, only the first character is used as the pad character.
  - Return type: String.

- **Description** The `SysStrPadRight` function (alias `padright`) returns a string whose length is `length` and:
  - if `length` is greater than the length of `string`, is `string` padded on the right with the `pad` character
  - if `length` is less than the length of `string`, is `string` truncated from the right to `length`.
  - if `length` is equal to the length of `string`, is `string`.

C.3.10 SysStrPadLeft, padleft

- **Synopsis**
  - `SysStrPadLeft(string,length,pad)`
  - `padleft(string,length,pad)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: Number.
  - Parameter 3 type: String. Although the type is String, only the first character is used as the pad character.
  - Return type: String.
- **Description** The `SysStrPadLeft` function (alias `padleft`) returns a string whose length is `length` and:
  - if `length` is greater than the length of `string`, is `string` padded on the left with the `pad` character
  - if `length` is less than the length of `string`, is `string` truncated from the left to `length`.
  - if `length` is equal to the length of `string`, is `string`.

C.3.11 SysFmtCurrTime, strftimecurr

- **Synopsis**

  - `SysFmtCurrTime(format)`
  - `strftimecurr(format)`
- **Parameters**
  - Parameter 1 type: String.
  - Return type: String.
- **Description** The `SysFmtCurrTime` function (alias `strftimecurr`) returns a string that represents the current time as formatted according to `format` using the C run-time `strftime()` function. Common values for `format` are:
  - %c - The preferred date and time representation for the current locale.
  - %d - The day of the month as a decimal number (range 01 to 31).
  - %F - Equivalent to %Y-%m-%d (the ISO 8601 date format).
  - %H - The hour as a decimal number using a 24-hour clock (range 00 to 23).
  - %j - The day of the year as a decimal number (range 001 to 366).
  - %m - The month as a decimal number (range 01 to 12).
  - %M - The minute as a decimal number (range 00 to 59).
  - %s - The number of seconds since the Epoch, 1970-01-01 00:00:00
  - %S - The second as a decimal number (range 00 to 60, allows for leap seconds).
  - %T - The time in 24-hour notation (%H:%M:%S).
  - %y - The year as a decimal number without a century (range 00 to 99).
  - %Y - The year as a decimal number including the century.
  - %% - A literal '%' character.
  - Any other characters, not specified by `strftime()`, are copied verbatim from `format` to the result string.

C.3.12 SysTime, time2epoch

- **Synopsis**
  - `SysTime(datetime,format)`
  - `time2epoch(datetime,format)`
- **Parameters**

  - Parameter 1 type: String.
  - Parameter 2 type: String. Default "%Y%m%d".
  - Return type: Number.

- **Description** The `SysTime` function (alias `time2epoch`) returns the number seconds since the Epoch calculated from `datetime` under the specification of `format`.

  The seconds since the Epoch, when interpreted as an absolute time value, represents the number of seconds elapsed since the Epoch, `1970-01-01 00:00:00 +0000 (UTC)`.

  `datetime` must be a string representation of a date and / or time and `format` must be a date format string that exactly describes `datetime` using the format characters as specified and used by the C function `strptime()`.

  Common options for the `format` are:

  - %% - The % character.
  - %c - The date and time representation for the current locale.
  - %C - The century number (0-99).
  - %d or %e - The day of month (1-31).
  - %H - The hour (0-23).
  - %I - The hour on a 12-hour clock (1-12).
  - %j - The day number in the year (1-366).
  - %m - The month number (1-12).
  - %M - The minute (0-59).
  - %p - The locale's equivalent of AM or PM. (Note: there may be none.)
  - %S - The second (0-60; 60 may occur for leap seconds; earlier also 61 was allowed).
  - %T - Equivalent to %H:%M:%S.
  - %x - The date, using the locale's date format.
  - %X - The time, using the locale's time format.
  - %y - The year within century (0-99). When a century is not otherwise specified, values in the range 69-99 refer to years in the twentieth century (1969-1999); values in the range 00-68 refer to years in the twenty-first century (2000-2068).

- %Y - The year, including century (for example, 1991).

C.3.13 SysStrFTime, strftime

- **Synopsis**
  - `SysStrFTime(seconds,format)`
  - `strftime(seconds,format)`
- **Parameters**
  - Parameter 1 type: Number.
  - Parameter 2 type: String.
  - Return type: String.
- **Description** The `SysStrFTime` function (alias `strftime`) returns a string date time representation of `seconds` formatted according to `format` as described in the C runtime function `strftime()`.

  `seconds` is the number of seconds since the Epoch, which when interpreted as an absolute time value, represents the number of seconds elapsed since the Epoch, `1970-01-01 00:00:00 +0000 (UTC)`.

  `format` must be a date format string used to format the returned date time string. For common values of `format` see section C.3.11 on page 56

C.3.14 SysInTable, intable

- **Synopsis**
  - `SysInTable(table,search)`
  - `intable(table,search)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Return type: Boolean.
- **Description**

  The `SysInTable` function (alias `intable`) returns a boolean `TRUE` if the value of `search` is found in the table of items `table`, otherwise it returns a boolean `FALSE`.

The value of `table` may be either the name of a text file in which each line is one element of the table, or a comma (,) or semi-colon (;) delimited string of the element values of the table.

- **Examples**
  - SysInTable("C:\customerNames.txt","Smith") This will test whether the name "Smith" occurs in the list of elements in the file `C:\customerNames.txt`.
  - SysInTable("/tmp/customerNames.txt",Record.Surname) This will test whether the name identified by the object types[13] field `Record.Surname` occurs in the list of elements in the file `/tmp/customerNames.txt`.
  - SysInTable("Smith,Jones,Right",Record.Surname) This will test whether the name identified by the object types[13] field `Record.Surname` occurs in the list of elements in the comma separated list specified by the first argument.

C.3.15 SysStrCondPack, condpack

- **Synopsis**
  - `SysStrCondPack(String,String)`
  - `condpack(String,String)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Return type: String.
- **Description**

  The `SysStrCondPack` function (alias `condpack`) returns a string which is conditionally formed by packing the string passed in the first parameter using the second parameter as a possible replacement character. If the first parameter matches the regular expression `X"[0-9][A-F][a-f]"` then the hexadecimal characters are packed into the corresponding encoding character set (ASCII or EBCDIC) characters. If the second parameter does not have a zero length, then the first character of this parameter string is used to replace all the non-graphic/non-printable characters of the packed character string. When the second parameter string has a zero length, then the character "?" is used as the replacement character for non-graphic/non-printable characters in the return string.

  If the first parameter string does not match the regular expression then the string is considered to already be packed. In this case, the string is still checked if the

second parameter length is greater than one and the non-graphic/non-printable characters are replaced by the first character of the second parameter string. When the second parameter string has a zero length, then the character "?" is used as the replacement character for non-graphic/non-printable characters in the return string.

- **Examples**
  - `condpack('X"414141"',"?")` on an ASCII based machine returns the string `AAA`.
  - `condpack('X"4141410000"',"?")` on an ASCII based machine returns the string `AAA??`.
  - `condpack("4141410000","?")` on an ASCII or EBCDIC based machine returns the string `4141410000`.

C.3.16 TermAppStructDataGet, sfget

- **Synopsis**
  - `TermAppStructDataGet(String,String)`
  - `sfget(String,String)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Return type: String.
- **Description** This function takes as the first parameter a value that should contain a `TermApp DE48-F0.16` Structured Data field and as the second parameter the name of a field within the structured data. The function will return the value of the named field as a string, if the name could not be found an empty string is returned.
- **Examples**
  - `sfget(DE48_FIELD,"OSVer")`
    Where**DE48_FIELD=**
    `219Postilion::MetaData275211FWSerialNbr11115SWRel11119CommsType11118TermType11115OSVer11116SWHash111211F01E201WSerialNbr2210100010000000102242315SWRel213144060219CommsType214INTERNAL MODEM 18TermType18EFTsmart15OSVer19820036078116SWHash18B4E1963A`

returns the string `820036078`

C.3.17 TermAppStructDataSet, sfset

- **Synopsis**
  - `TermAppStructDataSet(String,String,String)`
  - `sfset(String,String,String)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Parameter 3 type: String.
  - Return type: String.
- **Description**
- **Examples**
  - ```
    sfset(DE48_FIELD,"FWSerialNbr",
          "+----------LongerValue----------+")
    ```

    Where **DE48_FIELD** is initially set to

    ```
    219Postilion::MetaData275211FWSerialNbr11115SWRel111
    19CommsType11118TermType11115OSVer11116SWHash111211F
    WSerialNbr2210100010000000100224231 5SWRel2131401E201
    4060219CommsType214INTERNAL MODEM18TermType18EFTsmar
    t15OSVer1982003607816SWHash18B4E1963A
    ```

    Will return the updated value of **DE48_FIELD** as

    ```
    219Postilion::MetaData275211FWSerialNbr11115SWRel111
    19CommsType11118TermType11115OSVer11116SWHash111211F
    WSerialNbr233+----------LongerValue----------+15SWRe
    l2131401E2014060219CommsType214INTERNAL MODEM18TermT
    ype18EFTsmart15OSVer1982003607816SWHash18B4E1963A
    ```

C.3.18 gsub, replace

- **Synopsis**
  - `gsub(String,String,String,String)`
  - `replace(String,String,String,String)`
    ```

- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Parameter 3 type: String.
  - Parameter 4 type: String.
  - Return type: String.
- **Description** The function gsub() operates in much the same why as the `awk gsub` function does. The four parameters are
  1. Regular Expression (r) This parameter is a regular expression that should match one or more portions of the input text (t).
  2. Substitution String (s) This parameter is the replacement string
  3. Text to operate on (t) This parameter is the original input text value.
  4. How to operate (h) This parameter determines how many times the replacement text is substituted.

  How (h) can be either
  - g or G which means replace all occurrences of matched text with the substitution string.
  - Numeric which means replace only that occurrence.

  The regular expression (r) matches none, one or more portions of the input text (t) and based on the value of how (h) gsub() returns the input string where one or all of the matches are replaced with the substitution string (s).
- **Examples**
  - `gsub("a","bb",textfield,how)` This example specifies to replace the letter `a` with two letter `b`'s in `textfield` under the control of the variable `how`.

Textfield value	How	Returned Value	Description
abcdea12345a	G	bbbcdebb12345bb	Each a is replaced by two b's.
abcdea12345a	2	abcdebb12345a	The second a is replaced by two b's.
abcdea12345a	1	bbbcdea12345a	The first a is replaced by two b's.

Table 2: Effect of using gsub() to substitute text

  - `gsub("\([^ ]\+\) \([^ ]\+\)","\2 \1",textfield,how)` This example specifies to match two substrings that contain any character

except a space and that the first substring must be followed by a space followed by the second substring. The substitution string specifies to replace the whole matched value with the second matched substring followed by a space followed by the first matched substring. In other words it swaps two substrings around where the substrings do not contain a space and are separated by one space. The number of times the replacement is done is governed by the value of the variable `how`.

Textfield value	How	Returned Value	Description
ABC DEF	G	DEF ABC	The order of the two strings is reversed.
A1 bA1 A2 BA2	G	bA1 A1 BA2 A2	Each set of two strings are reversed.
A1 bA1 A2 BA2	2	A1 bA1 BA2 A2	Only the second set is reversed.

Table 3: Effect of using gsub() to substitute text

C.3.19 alias, lookup

- **Synopsis**
  - `alias(String,String)`
  - `lookup(String,String)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Return type: String.
- **Description** This function uses the second parameter as a lookup key to extract the associated value in the first parameter, which holds keyword value pairs. The value corresponding to the matched key word is returned. The keyword value pairs specified in the first parameter can either be a comma or semi-colon list of `keyword=value` pairs or a file name containing one `keyword=value` pair per line.
- **Examples**
  - `lookup("A=Alsatian,L=Labrador,S=Spaniel","L")`
    Will return the string "Labrador"
  - `lookup("D:/lookup.txt","L")`
    will return the string "Labrador" if the file `D:/lookup.txt` holds the following:

```
A=Alsatian
L=Labrador
S=Spaniel
```

C.3.20 uuid_time_gen

- **Synopsis**
  - `uuid_time_gen()`
- **Parameters**
  - Function takes no parameters.
- **Description** The function uuid_time_gen() uses the `libuuid` library (`https://sourceforge.` to generate a time based UUID using the system's local clock and network interface MAC address (if available). Each time the function is called it returns a new value which is expected to be globally unique.
- **Examples**
  - The sequence of calls to the `uuid_time_gen()` function returns a UUID formatted string:
  - `uuid_time_gen()` returns `a61293e0-58fe-11e8-97e1-f9f10e0b5eac`.
  - `uuid_time_gen()` returns `a6129462-58fe-11e8-97e1-f9f10e0b5eac`.
  - `uuid_time_gen()` returns `a612949e-58fe-11e8-97e1-f9f10e0b5eac`.
  - `uuid_time_gen()` returns `a61294d0-58fe-11e8-97e1-f9f10e0b5eac`.

C.3.21 pstore_set, psset

- **Synopsis**
  - `pstore_set(String,String,String)`
  - `psset(String,String,String)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Parameter 3 type: String.
  - Return type: String.

- **Description** This function sets a value in a persistent store specified in parameter 1 using the variable name specified in parameter 2 and the value in parameter 3. If an error occurs, for example not being able to connect to the persistent store server, an error condition is returned.

  The persistent store is identified by `host:port` where `host` is either an IP address or a DNS name that can be looked up and `port` is the port number on that host to which the persistent store server listens for incoming connections. The port (and the colon) can be left out in which case the default port is used. The default port is currently 60060.

- **Examples**
  - `pstore_set("www.codemagus.com:60069","ServerName","theCloud")` Will set and return the value of the variable `ServerName` to `theCloud` on the specified host.
  - `psset("www.codemagus.com:60069","ServerName","theCloud")` Will perform the same function as the example above.

C.3.22 pstore_get, psget

- **Synopsis**
  - `pstore_get(String,String)`
  - `psget(String,String)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Return type: String.
- **Description** This function retrieves a value from a persistent store specified in parameter 1 using the variable name specified in parameter 2. If the named variable is not found then an error condition is returned.

  The persistent store is identified by `host:port` where `host` is either an IP address or a DNS name that can be looked up and `port` is the port number on that host to which the persistent store server listens for incoming connections. The port (and the colon) can be left out in which case the default port is used. The default port is currently 60060.

- **Examples**
  - `pstore_get("www.codemagus.com:60069","ServerName")` Will return the value of the variable `ServerName` from the specified host.

  - `psget("www.codemagus.com:60069","ServerName")`
    Will perform the same function as the example above.

C.3.23 pstore_get_cset, psget_cset

- **Synopsis**
  - `pstore_get_cset(String,String,String)`
  - `psget_cset(String,String,String)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Parameter 3 type: String.
  - Return type: String.
- **Description** This function retrieves a value from a persistent store specified in parameter 1 using the variable name specified in parameter 2. If the named variable is not found then it is created with the default value specified in parameter 3 and that value is returned.

  The persistent store is identified by `host:port` where `host` is either an IP address or a DNS name that can be looked up and `port` is the port number on that host to which the persistent store server listens for incoming connections. The port (and the colon) can be left out in which case the default port is used. The default port is currently 60060.
- **Examples**
  - `pstore_get_cset("www.codemagus.com","ServerName","theNet")`
    Will return the value of the variable `ServerName` from the specified host (using the default port), but if it is not found will return and set it to `theNet`.
  - `psget_cset("www.codemagus.com","ServerName","theNet")`
    Will perform the same function as the example above.

C.3.24 pstore_get_incr, psget_incr

- **Synopsis**
  - `pstore_get_incr(String,String)`
  - `psget_incr(String,String)`

- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Return type: Number.
- **Description** This function retrieves a string representation of a numeric value from a persistent store specified in parameter 1 using the variable name specified in parameter 2. The numeric string is returned as a number type and is subsequently incremented by 1 and saved back to the persistent store as a numeric string.

  The persistent store is identified by `host:port` where `host` is either an IP address or a DNS name that can be looked up and `port` is the port number on that host to which the persistent store server listens for incoming connections. The port (and the colon) can be left out in which case the default port is used. The default port is currently 60060.
- **Examples**
  - `pstore_get_incr("www.codemagus.com:60069","Count")`
    If the value of `Count` on the persistent store is 3, then this function will return 3 and store 4 back on the persistent store. If the variable `Count` is not found an error condition is returned.
  - `psget_incr("www.codemagus.com:60069","Count")`
    Will perform the same function as the example above.

C.3.25 pstore_get_incr_cset, psget_incr_cset

- **Synopsis**
  - `pstore_get_incr_cset(String,String,Number)`
  - `psget_incr_cset(String,String,Number)`
- **Parameters**
  - Parameter 1 type: String.
  - Parameter 2 type: String.
  - Parameter 3 type: Number.
  - Return type: Number.
- **Description** This function retrieves a string representation of a numeric value from a persistent store specified in parameter 1 using the variable name specified

in parameter 2. The numeric string is returned as a number type and is subsequently incremented by 1 and saved back to the persistent store as a numeric string. If the named variable is not found on the persistent store then the default value specified in parameter 3 is returned and subsequently incremented and saved on the persistent store.

The persistent store is identified by `host:port` where `host` is either an IP address or a DNS name that can be looked up and `port` is the port number on that host to which the persistent store server listens for incoming connections. The port (and the colon) can be left out in which case the default port is used. The default port is currently 60060.

- **Examples**
  - `pstore_get_incr_cset("codemagus","Count",17)`
    If the value of `Count` on the persistent store is 3, then this function will return 3 and store 4 back on the persistent store. If the variable `Count` is not found then the value 17 is returned and 18 is saved to the persistent store as the value of `Count`.
  - `psget_incr_cset("codemagus","Count",17)`
    Will perform the same function as the example above.

C.3.26 runif

- **Synopsis**
  - `runif(Number,Number)`
- **Parameters**
  - Parameter 1 type: Number. Minimum value.
  - Parameter 2 type: Number. Maximum value.
  - Return type: Number.
- **Description** This function returns a uniform random number/deviate between the minimum value (first argument) and the maximum value (second argument). The minimum value must be strictly less than the maximum value. The random numbers returned are real numbers.
- **Examples**
  - `runif(1,100)` The call to function `runif` in this examples returns a uniform random real number in the range 1 to 100.

C.3.27 rnorm

- **Synopsis**
  - `rnorm(Number,Number)`
- **Parameters**
  - Parameter 1 type: Number. Mean value.
  - Parameter 2 type: Number. Standard deviation.
  - Return type: Number.
- **Description** This function returns a random/deviate number from a normal distribution with parameters supplied by the first and second parameters. The value supplied by the first argument is the population mean and the value supplied by the second argument is the population standard deviation. The standard deviation must be strictly greater than zero. The normal random number returned and the supplied mean value and standard deviation are real numbers.
- **Examples**
  - `rnorm(100,10)` The call to function `rnorm` in this examples returns a normal random real number with mean value if 100 and standard-deviation of 10.

C.3.28 rexp

- **Synopsis**
  - `rexp(Number)`
- **Parameters**
  - Parameter 1 type: Number. The mean value of the exponential distribution.
  - Return type: Number.
- **Description** This function returns a random number/deviate drawn from a negative exponential distribution. The value of the mean is supplied by the first and only parameter. The returned value and the value of the mean are real numbers.
- **Examples**
  - `rexp(20)` The call to function `rexp` in this examples returns an exponential random real number with mean value 20.

References

[1] recio: Record Stream I/O Library Version 1. CML Document CML00001-01, Code Magus Limited, July 2008. PDF.

[2] binary: Fixed and Variable Length Record Stream Access Method Version 1. CML Document CML00005-01, Code Magus Limited, July 2008. PDF.

[3] directory: Directory Record Stream Access Method Version 1. CML Document CML00014-01, Code Magus Limited, July 2008. PDF.

[4] MVS: MVS Record Stream Access Method Version 1. CML Document CML00016-01, Code Magus Limited, July 2008. PDF.

[5] remote: Remote Record Stream Access Method Version 1. CML Document CML00022-01, Code Magus Limited, July 2008. PDF.

[6] standard: Standard Input And Output Using Recio Version 1. CML Document CML00030-01, Code Magus Limited, July 2008. PDF.

[7] text: File Access Method Using POSIX Streams Version 1. CML Document CML00031-01, Code Magus Limited, July 2008. PDF.

[8] image: DB2 Image Copy Reader Access Method Version 1. CML Document CML00036-01, Code Magus Limited, July 2008. PDF.

[9] edit: Recio Edit Access Method Version 1. CML Document CML00047-01, Code Magus Limited, June 2009. PDF.

[10] db2query: Recio DB2 Query Access Method Version 1. CML Document CML00050-01, Code Magus Limited, November 2009. PDF.

[11] db2dclgen: Recio DB2 DCL Generator (DCLGN) Access Method Version 1. CML Document CML00051-01, Code Magus Limited, November 2009. PDF.

[12] Code Magus Limited. Symbol Table Loading From Copybooks. CML Document CML00039-01, Code Magus Limited, July 2008. PDF.

[13] Code Magus Limited. objtypes: Configuring for Object Recognition, Generation and Manipulation. CML Document CML00018-01, Code Magus Limited, July 2008. PDF.