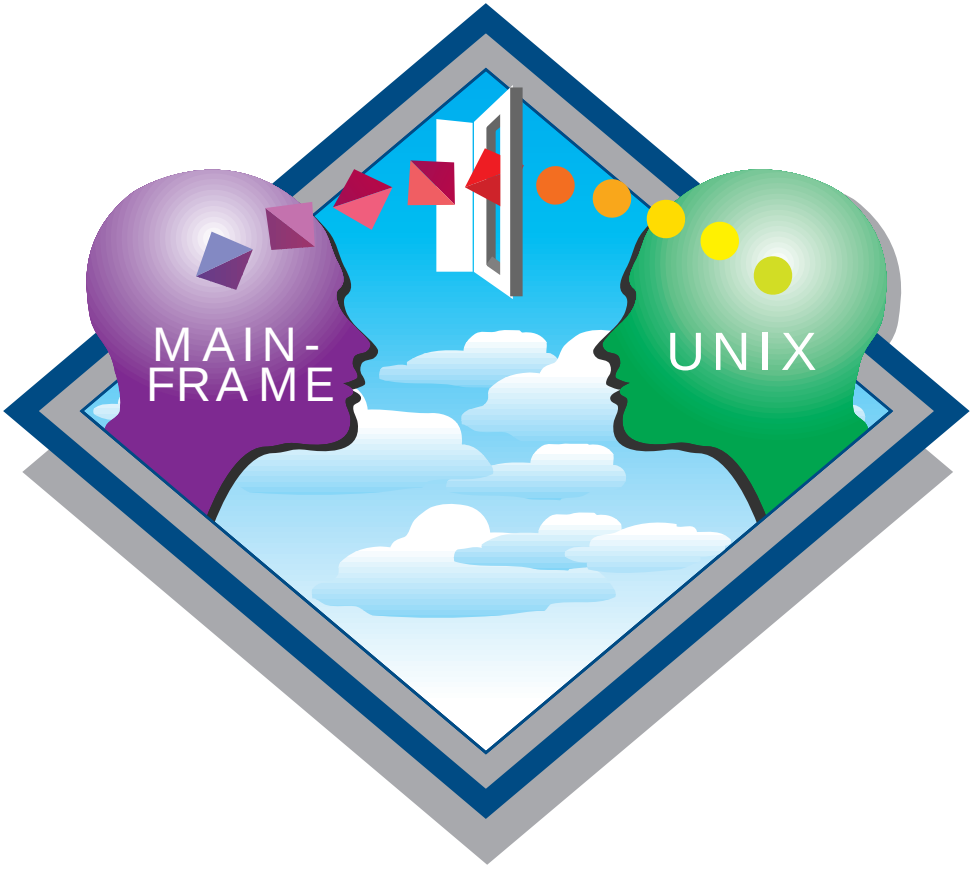
FilePort

Release 3

User's Guide



syncsort

SI-3004-2
3.0 v2

Table of Contents

Part II. Converting from UNIX to Mainframe

Appendices

Chapter 1. Introduction

FilePort converts IBM mainframe data sets to UNIX files, and UNIX files to IBM mainframe data sets. The data to be converted can contain any combination of character, decimal, packed decimal and binary fields.

Processing Overview

FilePort takes a data file and one or more record layouts as input.

It uses the record layouts to distinguish between different fields in the records and converts the data according to the type of each field (character, decimal, packed decimal, binary, etc) and the machine architecture of the UNIX system (big-endian or little-endian).

The record layouts can be passed in existing COBOL copybooks, or defined through FilePort options.

If the records in the input file do not all have the same layout, different record layouts can be supplied, and conditions can be specified to determine which layout to use for each record.

FilePort writes the converted records to an output file.

What this Manual Contains

This user guide has two parts plus appendices:

Part I (Chapter 2 through Chapter 5) describes converting mainframe data sets to UNIX files. **Part II** (Chapter 6 through Chapter 10) describes converting UNIX files to mainframe data sets. These two functions can be licensed as independent units, so your specific installed copy of FilePort may perform both functions or just one of the functions, depending on your software license.

With the exception of Chapter 6, chapters in Part I and Part II are parallel:

- Chapters 2 and 7 describe input to FilePort.
- Chapters 3 and 8 describe output from FilePort.
- Chapters 4 and 9 describe the FilePort command and options. Note that the same command and options are used for translation in both directions; the separation is purely for simplicity of presentation, and conversion of mainframe-to-UNIX applications to the equivalent UNIX-to-mainframe applications is relatively simple (see chapter 6).
- Chapters 5 and 10 provide examples.
- Chapter 6 is unique to the UNIX-to-mainframe direction. It describes procedures for changing mainframe-to-UNIX applications to UNIX-to-mainframe.

Appendices contain information common to both forms of FilePort:

- Appendix A describes the various record formats handled by FilePort on input and output.
- Appendix B describes the mainframe and UNIX data types, and the conversions FilePort performs.
- Appendix C gives the translation tables used by FilePort for conversion of character data from EBCDIC to ASCII and vice versa.
- Appendix D describes issues specific to COBOL including field alignment, etc.
- Appendix E contains details of how to refer to field names.
- Appendix F describes the message utility, which gives online explanations and actions for all messages generated by FilePort.
- Appendix G contains information to enable you to contact Syncsort's Technical Support group, should you have any questions or problems.

Part I. Converting from Mainframe to UNIX

Chapter 2. Mainframe Input to FilePort

Mainframe Data Set

FilePort takes a single sequential mainframe data set as input.

The data set can be on IBM standard labeled or unlabeled tape, in a UNIX file system, or piped through standard input.

The data set format can be IBM MVS or VSE sequential with fixed length (blocked or unblocked) or variable length (blocked and optionally spanned) records (F, FB, V, VB, VBS). To convert a VSAM data set to UNIX, use the mainframe SyncSort COPY or IDCAMS REPRO feature to convert it to a sequential data set before moving it to UNIX.

The location and format of the data set are specified through the **-infile** option described in Chapter 4.

Data Sets Moved on Tape

If the data is moved to UNIX on tape, the information you need to specify to FilePort depends on whether the tape has standard IBM labels or not.

If the tape has standard IBM labels, specify the **labeled** argument on the **-infile** option and FilePort will then pick up blocking information from the labels. If the tape contains multiple data sets, specify either the **DSNAME=** argument or the **filenumber** argument to tell FilePort which data set to convert.

If the tape is unlabeled, use the **blocksize** argument on **-infile** to specify the blocksize used when the tape was created. If the tape contains multiple data sets, specify the **filenumber** argument to tell FilePort which data set to convert. If the data set spans multiple tapes, specify the **multivolume** argument to tell FilePort how many tape volumes to read.

Data Sets Moved by ftp

If the data is moved to UNIX through a network, using **ftp** or an equivalent function, specify BINARY as the transfer type to **ftp**, so that data is not translated during transfer.

If the data contains variable length records and the LOCSITE RDW and BINARY options are specified to **ftp**, specify the **unblockedvariable** record format. Alternatively, if the data contains variable length records and the LOCSITE RDW option is not specified, the **ftpbinary** record format argument on the **-infile** option should be specified.

Record Layout

The input data records must be described to FilePort in order for it to convert individual data fields. This record layout can be provided through a COBOL copybook (**-datadictionary** option), or through a FilePort **-recordlayout** option. If neither **-datadictionary** nor **-recordlayout** is specified, FilePort treats the whole record as character data. If the records in the input file do not all have the same layout, conditions can be specified to determine which layout to use for each record (**-condition** option).

Record Layout in a COBOL Copybook

If the layout of the records in your input file is available in a separate copybook, FilePort can read it directly when you use the **-datadictionary** option described in Chapter 4.

The translation of field values by FilePort is controlled mainly by the USAGE, PICTURE and SIGN clauses in the data description. Each elementary data item, either described explicitly by a data-name or as a filler, is converted. The relationship between COBOL data types and FilePort data types is described in the **-datadictionary** option in Chapter 4 and in the data type description in Appendix B.

The copybook itself must be available as a standard text file (ASCII text, records terminated by linefeeds) on the UNIX system. If you move the file to UNIX across a network (using **ftp**, for example), the text may be

converted automatically. If the file is still in mainframe format, you can use FilePort to convert it to UNIX format:

```
fileport -i input_file -o output_file linefeed
```

where *input_file* is the name of your EBCDIC copybook and *output_file* is the name you want to give to the ASCII version.

Record Layout in a FilePort Option

When the record layout of your input file is not available as a COBOL data description file, you must define the positions and formats of the fields through the **-recordlayout** option described in Chapter 4.

Multiple Record Layouts

When your records do not all have identical layouts, use **-datadictionary** and/or **-recordlayout** options to describe all the possible layouts. Then use a **-condition** option for each different layout to define which records have that layout. The **-condition** option defines a condition of the input data records (e.g., the **record-type** field has the value **customer**) and specifies which layout to use for records of that type.

Chapter 3. UNIX Output from FilePort

FilePort translates fields in your input records according to the record layout you supplied, and generates a sequential output file. FilePort picks defaults for the record format and the output data types. You may specify options to change some of the defaults.

Default Conversions

Record Format

The default record format generated by FilePort is fixed for fixed input (F, FB), and binary variable for variable input (V, VB, VBS). All records on UNIX are automatically blocked and the blocking factor is not user-specifiable. Record formats are described in detail in Appendix A.

Output Data Type

The default data types generated are the UNIX equivalents of the mainframe data types, i.e. character data and display numeric data become ASCII rather than EBCDIC, integer data is output unchanged on big-endian systems and converted to little-end format on little-endian systems, packed decimal data is output unchanged, and floating point data becomes IEEE format rather than IBM excess-64 format. Embedded sign data is output in Micro Focus COBOL format. Details of the machine representations of the various data types and their conversions are given in Appendix B.

Slack bytes which exist in the record as the result of field alignment (e.g. through the COBOL SYNCHRONIZED clause) are left in the output record.

Changing Default Conversions

You can use command options to specify the following changes to the default output from FilePort. See the **-outfile** option in Chapter 4 for details.

Text Record Format

Records can be converted to standard UNIX text records, for example, for input to a database load utility or to a spreadsheet. A UNIX text file contains records with all data in display format, fields optionally separated by a user-defined special character (comma, space, etc.), fields optionally compressed by removing spaces and non-significant zeroes and each record terminated by a linefeed character.

Use the **display** or **coboldisplay** arguments to the **-outfile** option to convert the data to displayable format; the **separate** argument to generate field separators; the **compress** argument to remove leading and trailing blanks from fields; and the **linefeed** argument to generate linefeed terminators. Alternatively, you can specify the **text** argument, which has the same effect as **linefeed display separate compress noslack**.

Micro Focus Variable Record Format

Records can be converted to Micro Focus variable record format. Use the **mfvariable** argument to the **-outfile** option.

MS-DOS Text Record Format

Records can be converted to MS-DOS text records, where each record is terminated by a carriage return and a linefeed. Specify the **crlf** argument to the **-outfile** option.

FORTRAN Unformatted Record Format

Records can be converted to standard FORTRAN unformatted records with a 4 byte record descriptor before and after each data record. Use the **fortran** argument to the **-outfile** option.

Zoned Decimal and Embedded Sign Data

If your mainframe data definitions specify COBOL DISPLAY usage, a signed numeric PICTURE clause (containing S and one or more of 9, P, V), and no SEPARATE SIGN clause, then your records will contain embedded sign data (also known as zoned decimal data). Different UNIX COBOL compilers have different ways of representing embedded sign data. By default, FilePort translates embedded sign decimal data (mainframe zoned decimal) to the Micro Focus COBOL format. You can request that the output types be changed to an alternative format (used for example by RM COBOL and DEC COBOL) by specifying the **altzoned** argument on the **-outfile** option. Details of the different formats are given in Appendix B.

Removal of Slack Bytes

If your mainframe data definition uses the COBOL SYNCHRONIZED clause then your records may contain slack bytes, inserted by the compiler in order to align specific data fields on word boundaries. If you want the slack bytes removed in the UNIX version of the file use the **noslack** argument to the **-outfile** option. Slack bytes are automatically removed if you specify **text** as the output record format. See Appendix D for more information on COBOL slack bytes and the behavior of COBOL compilers.

Chapter 4. Mainframe-to-UNIX Command Reference

The fileport Command

You use the **fileport** command to invoke FilePort from the UNIX shell prompt.

The format of the command is:

fileport *option ...*

An *option* consists of a keyword (one of **-condition** , **-datadictionary, -infile** , **-outfile** , **-recordlayout** , **-warning**) plus arguments to the keyword. You can provide the options in any order. The options are described in the following sections.

The following typographical conventions are observed in the format statements of FilePort command options.

- Variables are shown in *italics*. You must substitute an appropriate numerical or text value for a variable.
- Constants are shown in **bold** text and must be specified as shown. You may supply a constant keyword in either lower or upper case and you may abbreviate it (e.g., **-infile, -o, ALTZ, fixed**).
- Braces { } indicate that a choice must be made from the set of items contained in the braces.
- Brackets [ ] indicate that an item is optional.

- An ellipsis ... indicates that the preceding argument or group of arguments may be repeated.

Correction of Options Problems

When FilePort finds problems in the options, you are notified of each problem in turn. Up to four pieces of information are displayed for each problem.

- The line of the option containing the problem.
- An indicator pointing to the position in the line at which the problem was detected.
- A descriptive message preceded by a mnemonic in parentheses.
- If the error was found in the contents of a COBOL data description file, a message indicating which line in the file is in error.

You may use the message mnemonic with the Syncsort Message Utility, syncmsg, to get more information on the exception. See Chapter 11 for a full description of the syncmsg utility.

Completion Status

After you run FilePort, the value of the shell variable **?** indicates the success or otherwise of the conversion.

0 FilePort completed successfully.

100 FilePort completed but output generated may be incorrect or incomplete, e.g. due to the values of the input data not matching the specified data type.

111 FilePort did not complete successfully.

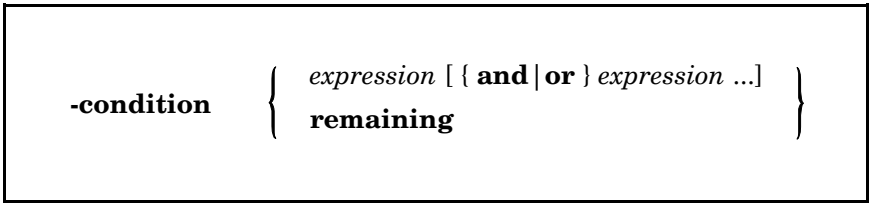
You can check the completion status of FilePort by typing the UNIX command **echo $?** (in Bourne shell or Korn Shell) or **echo $status** (in C-shell) after FilePort has executed.

-condition

Purpose

To define a condition for selecting individual records based on the contents of one or more fields.

Format



where

expression = *field_name* { **eq** | **ne** | **lt** | **gt** | **le** | **ge** } { *field_name* | [*repeat*] *string* | *number* }

For use of parentheses within an expression, see the notes below.

Arguments

field_name the name of a field defined in a COBOL copybook or in a **-recordlayout** option. The field must be uniquely identifiable from all the record layouts. Use qualified field names, described in Appendix E, to distinguish between fields of the same name. Fields used in **-condition**

expressions can have any data type other than floating point.

repeat — the number of times the *string* is to be repeated to yield the value used in the expression.

string — a character string constant enclosed in double quotes. To include a double quote in a string, use two double quotes. Since UNIX shells may strip double quotes from the command line, you will generally need to enclose pairs of double quotes within pairs of single quotes. A string can be compared only to a field of type character.

number — a numeric constant, consisting of an optional sign followed by one or more digits with an optional decimal point. A number can be compared only to a numeric field.

Location

Each occurrence of the **-condition** option must be immediately followed by a **-recordlayout** or a **-datadictionary** option. If any conditions are defined, the last **-condition** option must be **-condition remaining** .

Notes

Each condition is evaluated in sequence for each input record. When a record satisfies a condition, it is assumed to have the layout in the **-datadictionary** option or the **-recordlayout** option immediately following that condition. Further **-condition** options are not checked for that record.

The **remaining** condition is satisfied by all records that do not satisfy any preceding **-condition** option. This must be the last condition option that appears in an application.

You may build compound logical expressions by using **and**s, **or**s and pairs of parentheses. Expressions within parentheses are evaluated first. When the

order of evaluation is not indicated by parentheses, **and** is evaluated before **or**. Multiple **and** or **or** operators are evaluated from left to right.

Examples

```
-condition  state-code eq '"CA"' or state-code eq '"NY"'

-condition  (status eq 14 or status eq 15) and
            amount gt 99999.99

-condition  remaining
```

-datadictionary

Purpose

To specify the location of one or more input record layouts held in a COBOL copybook.

Format

-datadictionary *file_name* [*field_name* [, *field_name* ...]]

Arguments

file_name the UNIX pathname of a file which contains one or more complete COBOL data descriptions with level 01 entries which define the input record layout(s).

field_name the name of a field defined in the COBOL data descriptions. The field name must be uniquely identifiable. Use fully qualified field names to distinguish between fields of the same name. See Appendix E for information on how to specify qualified field names.

Location

This option may appear anywhere in the options definition if you have a single record layout for all the records in your input. If you have multiple layouts for your input records, each **-datadictionary** option must appear immediately following the corresponding **-condition** option. See the section on the **-condition** option in this chapter for details.

Notes

Defaults

If neither **-datadictionary** nor **-recordlayout** is specified, FilePort treats each record as all character data.

COBOL Language Elements

FilePort supports a data description written to the specifications for Release 4 of *VS COBOL II*. Table 1 on the next page summarizes the various data categories available through *VS COBOL II* and indicates the equivalent FilePort data type for each. Note that not all IBM extensions to the standard *X3.23-1985, American National Standard for Information Systems - Programming Language - COBOL* are supported for translation.

Level 66 (RENAMES), level 77 and level 88 data entries are ignored.

The BLANK WHEN ZERO, EXTERNAL, GLOBAL and VALUE clauses are ignored.

A symbolic character is not supported as a figurative constant.

Multiple Record Layouts

A COBOL copybook may contain multiple record layouts in the form of multiple 01 level data items, or data items qualified by the REDEFINES clause.

When the copybook holds multiple record layouts, you can specify *field_name* argument(s) to provide a unique record layout to FilePort.

COBOL Data Type				FilePort Data Type
Category	**Picture**	**Usage**	**Sign**	
alphabetic	A	[DISPLAY]	none	character
numeric	9 P V	[DISPLAY]	none	decimal, unsigned
numeric	S 9 P V	[DISPLAY]	none	decimal, embedded sign, trailing
numeric	S 9 P V	[DISPLAY]	LEADING	decimal, embedded sign, leading
numeric	S 9 P V	[DISPLAY]	LEADING SEPARATE	decimal, separate sign, leading
numeric	S 9 P V	[DISPLAY]	TRAILING SEPARATE	decimal, separate sign, trailing
numeric	9 P V	BINARY, COMP, COMP-4	none	binary, unsigned integer (see also Appendix B)
numeric	S 9 P V	BINARY, COMP, COMP-4	none	binary, signed integer (see also Appendix B)
numeric	S 9 P V	PACKED-DECIMAL, COMP-3	none	packed decimal
numeric edited	S 9 P V B Z * 0 / , . + - CR DB	[DISPLAY]	none	character
alphanumeric	X	[DISPLAY]	none	character
alphanumeric edited	A X 9 B 0 /	[DISPLAY]	none	character
DBCS	G B	DISPLAY-1	none	**not supported**
external floating point	9 V . E + -	[DISPLAY]	ignored	**not supported**
floating point	none	COMP-1, COMP-2	none	binary, floating point
index	none	INDEX	none	binary, signed integer
pointer	none	POINTER	none	binary, signed integer

Table 1. Data Categories Available Through VS COBOL II

Specify a *field_name* for each data item needed to distinguish between alternative layouts, as follows:

- to choose between complete data descriptions in the copybook, specify the level 01 data item name associated with the desired data description.

- to choose between redefined portions of the data description, specify the name of a data item included in one of the redefinitions. Designating a data item implicitly designates any group item of which it is a subordinate.

Therefore, you cannot specify data items which are subordinate to two group items that redefine each other. For example, you cannot specify two items subordinate to different level 01 items.

When you identify a redefining item of the data description as part of the record layout and it is shorter than the redefined item, the portion of the record not covered by the redefining item is unchanged in the output.

When you do not provide information to distinguish between layouts, FilePort uses the first of any set of choices, as follows:

- when the copybook contains multiple level 01 data items, FilePort chooses the first one.

- when a subordinate data item has a REDEFINES clause, FilePort chooses the redefined item over the redefining item.

A level 66, level 77 or level 88 data item cannot be specified as the *field_name* argument.

When the input record format is fixed, all the record layouts must have the same length, or the record length must be specified in the **–infile** option.

When the input record length is specified in the **–infile** option, and the length of any record layout is longer than the specified length, FilePort stops with an error. If any record layout is shorter than the specified length, the remainder of the record is unchanged in the output.

Examples

```
-datadictionary company.cbl

-datadictionary /general/production/masterfile.cob

-datadictionary copybook_1 level2-BC
```

-infile

Purpose

To define the source of data records to be converted.

Format

-infile *file_identifier* [*record_format*] [*max_length*]
[**blocksize** *block_size*]
[**labeled** [**"DSN**[**AME**]**=***dataset_name***"**]]
[**filenumber** *file_number*]
[**multivolume** *num_volumes*]

where

file_identifier = { *file_name* | *file_descriptor* **open** }

record_format = { **fixed** | **unblockedvariable** | **variable** | **ftpbinary** }

Arguments

file_name the name of a file which contains records to be converted

file_descriptor	the file descriptor of an open file which contains records to be converted.
max_length	the length in bytes of the records in the file (for fixed records) or the longest record in the file (for variable records). The length for variable records includes the length of the RDW, for example if your record contains 100 bytes of data your *max_length* will be 104.
block_size	the block size in bytes for an input file on tape.
file_number	the file number on tape, starting at 1, when you have a tape that contains multiple data sets.
num_volumes	the number of tape volumes to be read. Required if you have multiple volumes of an unlabeled tape file.
dataset_name	name of the data set on the labeled tape(s) that you want to use as input. The data set name can contain letters, digits, '$', '#', '@', and '-'. The first character cannot be a '-' or a digit. If the data set name contains any characters other than letters and digits, enclose it within a pair of single quotes. This option overrides the **filenumber** option.

Notes

The input to FilePort is a data set on disk, tape, or in standard input, which contains the data you want to convert. Typically, the data set will contain records with a mixture of character data and packed decimal or binary numeric data.

Input Device

Disk. Provide the UNIX file name as the *file_name* argument.

Tape. Provide the tape device file name as the *file_name* argument. The tape must be mounted and positioned to the start before executing FilePort.

The tape may be unlabeled, or have standard IBM labels. If the tape is labeled, or if you have multiple files on the tape and the file you want is not

the first one, then you must provide a device name that is both non-rewinding and allows multiple files to be read from tape. Non-rewinding tape device files typically contain an **n** in their name, and device files allowing multiple files have a **b** in their name on some systems, e.g. **/dev/rmt/0mbn**. If in doubt about which device name to use, consult your system administrator.

If the tape is labeled, specify the keyword **labeled**. FilePort then reads information like the block size, the record format, etc. from the tape label(s). You can override any information in the label(s) by specifying command line arguments. For an unlabeled tape, you must provide the block size that was used when writing the tape, through the **blocksize** *block_size* argument.

If you have multiple files on the same tape, specify the **filenumber** argument or the **DSNAME** label option to identify which data set you want to convert. FilePort scans all the volumes of a multiple volume group until it finds the right data set. [The DSNAME option may not be available for all platforms. Refer to your release-notes for information on supported platforms.]

If your tape file is unlabeled and occupies multiple volumes, specify the number of volumes to FilePort using the **multivolume** argument. [Multivolume tape support may not be available for all platforms. Refer to your release-notes for information on supported platforms.] If your tape file is labeled and occupies multiple volumes, FilePort will automatically read the correct number of tapes. You can override the number of volumes read for a labeled tape by specifying the **multivolume** argument. For labeled tapes, FilePort also verifies the data set sequence number in the data set headers on the tapes to make sure the tapes are mounted in the correct sequence.

For multivolume input, FilePort expects each volume to be mounted on the same tape drive. At the end of each tape volume, FilePort displays a message on *stderr* and polls the tape drive, waiting for the next tape to be made online. If the input device is an autoloader, the next volume should automatically get mounted at this point. Otherwise, mount the next tape in the drive.

Standard input. Specify the *file_descriptor* as 0, with the **open** keyword and record information. For example, **-infile 0 open fixed 80** .

Record Format and Length

The records in the input file may be in either **fixed** or **variable** format, as supported under the mainframe file system. If you use **ftp** to transfer files from the mainframe to UNIX over a network, the following additional formats are supported:

unblockedvariable this format should be specified when the file is transferred using the **ftp** LOCSITE RDW option.

ftpbinary this format should be specified when a file is transferred using the **ftp** BINARY option.

The default record format for disk, unlabeled tape and *stdin* input is determined by the record layout definition. If the record layout definition is defined using a COBOL copybook (see **-datadictionary** option) and the copybook contains **variable** length arrays (using 'OCCURS DEPENDING ON' clause), the default record format is **variable**. Otherwise, the default record format is **fixed**. For labeled tapes, the record format and maximum record length are read from the tape header when not specified in the command options.

Examples

```
-infile  mainframe_input_file

-infile  mainframe.to.unix variable 82

-infile  tape.in fixed 128 blocksize 512

-infile  0 open variable 200

-infile /dev/rmt/0m label '"DSNAME=PRDABC.SEQ1.TAPE"'
```

-outfile

Purpose

To specify the destination of converted records and any special conversions required.

Format

-outfile *file_identifier* [*disposition*] [**blocksize** *block_size*] [*record_format] [max_length [min_length]]* [*field_option*] [*data_conversion*]

where

file_identifier = { *file_name* | *file_descriptor* **open** }

disposition = { **overwrite** | **append** }

record_format = { **crlf** | **fixed** | **fortran** | **linefeed** | **mfvariable** | **text** | **variable** }

field_option = [**separate** [*field_separator*]] [**compress**] [**noslack**] [**occursdependingon**] [*sequence_name*] **[padspace]**

field_separator = { "*a*" | **x**"*xx*" | **o**"*ooo*" }

data_conversion = { **altzoned** | **coboldisplay** | **display** }

Arguments

file_name	the name of a file which is to receive converted records
file_descriptor	the file descriptor of an open file which is to receive converted records
max_length	the length in bytes of the longest record in the file
min_length	the length in bytes of the shortest record in the file
a	the character value of the field separator character.
xx	the hexadecimal value of the field separator character.
ooo	the octal value of the field separator character.
block_size	the block size in bytes for a tape output file.
sequence_name	the name of the translation table. See Appendix C for all acceptable values for the sequence name argument.

Notes

FilePort writes the records after translation to an output file, which can be on disk or tape, or to standard output.

Output Device

Disk. FilePort uses the name you provide in the *file_name* argument.

Tape. Provide the tape device file name as the *file_name* argument. The tape must be positioned to the start before executing FilePort.

For multivolume output, FilePort expects each volume to be mounted on the same tape drive. At the end of each tape volume, FilePort displays a message on *stderr* and polls the tape drive, waiting for the next tape to be made online. If the output device is an autoloader, the next volume should automatically get mounted at this point. Otherwise, mount the next tape(s) in the drive. [Multivolume output may not be available for all platforms, refer to your release-notes for information on supported platforms.]

Standard output. When you wish FilePort to supply the output records to standard output, specify the file descriptor of standard output and the **open** keyword, plus any record information. For example, **-outfile 1 open** .

File Organization

The converted records are written to a file with sequential organization.

The order of records in the output file is the same as the order of records in the input.

File Disposition

If the file does not already exist, a new file is created. If you wish to use an existing file, you must specify either **overwrite** or **append**.

overwrite specifies that if the output file already exists, the contents are to be replaced by new output.

append specifies that if the output file already exists, new output is to be appended to the end.

Record Alignment

The converted records are not aligned, i.e. each record starts in the byte immediately following the last byte of the previous record.

Record Format

The default output record format is fixed for fixed input, and variable for variable input. If you specify the **separate** argument as a field option, then the format defaults to **linefeed**. A detailed description of record formats is given in Appendix A.

Record Lengths

If minimum length is specified, maximum length must also be specified. If maximum and minimum lengths are specified and record format is **mfvariable**, the maximum and minimum lengths are recorded in the Micro Focus file header.

Field Options

Field alignment. The converted output data fields are not aligned, i.e. each data field starts in the byte immediately following the last byte of the previous field, unless your input record layout is from a COBOL copybook and contains a SYNCHRONIZED clause.

In the case that your input record layout is from a COBOL copybook and contains a SYNCHRONIZED clause and you want to make sure that the output data fields are not aligned, i.e. not separated by any slack bytes, then specify the **noslack** argument. This should allow you to use the output file with a COBOL program and utilize the same COBOL copybook file there that you provided to FilePort through the **-datadictionary** option. A detailed description of COBOL slack bytes is given in Appendix D.

Field separators. By default, fields are output adjacent to one another. If you want them to be separated, for example by a space, a tab, a comma, etc., then use the **separate** argument. If you specify the **separate** argument, FilePort will insert a field separator between each elementary field in the **-recordlayout** option or the COBOL copybook. If the field separator character you want is other than the UNIX default (a blank), specify it through the *field_separator* argument. Since UNIX shells may strip double quotes from the command line, you will generally need to enclose the double

quotes around the separator within a pair of single quotes. When you specify the **separate** argument, the record format defaults to **linefeed**.

Field compression. If you specify the **compress** argument, FilePort will trim all leading and trailing spaces from each character field and all leading spaces and leading zeroes from each edited numeric field. A single zero remains in a numeric field when the field contains a value zero. When you specify the **compress** argument, **separate** is implied.

Numeric field padding. By default, FilePort pads the converted numeric fields with leading zeros if needed when the **display** option is specified. FilePort will pad the converted numeric fields with leading spaces rather than zeros if the **padspace** option is set. The **padspace** option is ignored if the **display** option is not set.

Occursdependingon. By default, FilePort expands any variable length array defined by an OCCURS DEPENDING ON clause to the maximum size.

If you would like the array to remain variable, specify the **occursdependingon** argument. If the output file generated using **occursdependingon** is going to be used by a Micro Focus COBOL program, you will have to use the ODOOSVS or the ODOSLIDE compiler option.

Sequence name. By default, FilePort uses the first EBCDIC to ASCII translation table in Appendix C for mainframe to UNIX data conversion. Alternatively, other sequence names may be specified. Table 1 on page C-1 lists these sequences and gives the page number in the appendix where the associated translation tables may be found.

Data Type Conversion

Each mainframe data field is converted to its appropriate UNIX format according to the rules detailed in Appendix B. The conversion is modified when you specify one of the following three arguments.

Display. If you specify the **display** keyword, FilePort converts all numeric fields to displayable form, i.e. embedded sign decimal fields, packed decimal fields and binary fields are converted to edited numeric data type. Binary floating point fields are not currently converted to display.

Coboldisplay. If you specify the **coboldisplay** keyword, FilePort converts all numeric fields to the format used by the DISPLAY statement of a

COBOL program. Binary floating point fields are not currently converted to this format. The **coboldisplay** keyword is incompatible with the **compress**, **separate**, and **text** keywords.

Altzoned. If your records contain display numeric decimal data with an embedded or overpunched sign, also referred to as zoned decimal data, then the output data can be translated to one of two formats. By default, FilePort translates to the Micro Focus format. If you want the other format, specify the **altzoned** keyword. Details of the two formats are shown in Appendix B.

Examples

```
-outfile unix.out

-outfile fieldsep.out separate '";"' compress overwrite

-outfile mf.output mfvariable noslack

-outfile op.output altzoned variable

-outfile 1 open fixed

-outfile unix_columns separate display

-outfile unix_format text

-outfile unix.out padspace display
```

-recordlayout

Purpose

To describe the layout of the input records.

Format

-recordlayout *field* [, *field* ...]

where

field = { *elementary_field* | *composite_field* }

elementary_field = { *fieldname* [*length* [*datatype*] [**repeat** *count*]] | **filler** *length* }

The first format is used to describe a field which is to be converted. The second format is used to allow for portions of the input record which do not need to be converted.

composite_field = *fieldname* **{** *field* [, *field* ...] **}** [**repeat** *count*]

Note that the braces in *composite_field* must appear where shown. They do not indicate a choice of arguments in this case.

Arguments

fieldname the name of a composite or elementary field in the input record layout. The name assigned to a field must adhere to the rules for an identifier defined in Appendix E. The name of a field cannot be the same as the name of an including composite field.

count the number of times a field is repeated consecutively in the input record.

datatype the type of data held in a field. The default data type is character.

length the length of a field, in bytes. The length may be omitted only for a variable length character field at the end of the input record.

Location

This option may appear anywhere in the options definition if you have a single record layout for all the records in your input. If you have multiple layouts for your input records, each **-recordlayout** option must appear immediately following the corresponding **-condition** option.

Notes

Defaults

If neither **-datadictionary** nor **-recordlayout** is specified, FilePort treats each record as all character data.

Data Types

The **-recordlayout** option allows you to specify your data fields as any of the standard data types as described in Appendix B. If your record contains a field with a non-standard data type, you can specify that part of the record to FilePort as a **filler**. The data will be unaffected by the translation process and will appear in the output file in the same form as in the input file.

Table 2 on the next page summarizes the supported data types and gives the keyword and range of lengths to be used in the **-recordlayout** option for each type.

FilePort Data Type	Keyword	Length (bytes)
binary, floating point	FLOAT	2 -16
binary, signed integer	INTEGER	1 - 256
binary, unsigned integer	UINTEGER	1 - 32,767
character	CHARACTER	1 - 32,767
decimal, edited numeric	EN	1 - 256
decimal, embedded sign (zoned decimal)	LP, TP	1 - 256
decimal, separate sign	LS, TS	2 - 256
decimal, unsigned	AN	1 - 256
filler	FILLER	1 - 32,767
packed decimal	PD	1 - 256

Table 2. Input Data Types and Lengths Available through ***-recordlayout***

Elementary, Composite and Repeated Fields

The basic building block of the record layout is an *elementary field*, which describes one contiguous portion of data in an input record. An elementary field is not subdivided further. In many cases, your input record layout will consist simply of a list of elementary fields. When an elementary field in the input record is repeated several times in the record without any intervening fields or fillers, you can conveniently specify the field only once and add the **repeat** argument to indicate how many times it is repeated.

When two or more consecutive elementary fields in your input record are repeated several times as a block, it is convenient to consider the repeated elementary fields as a single larger field. A field which is made up of several elementary fields is referred to as a *composite field*. For example, a composite field **full_name** may consist of three elementary fields, **first_name**, **middle_initial** and **last_name**. A composite field can have other composite fields as components, as well as elementary fields.

The elementary fields which make up a composite field do not have to have the same data types. You cannot assign a data type to a composite field.

Multiple Record Layouts

When the input record format is fixed, all the record layouts must have the same length, or the record length must be specified in the **-infile** option.

When the input record length is specified in the **-infile** option, and the length of any record layout is longer that the specified length, FilePort stops with an error. If any record layout is shorter than the specified length, the remainder of the record is treated as a filler, and is unchanged in the output.

Examples

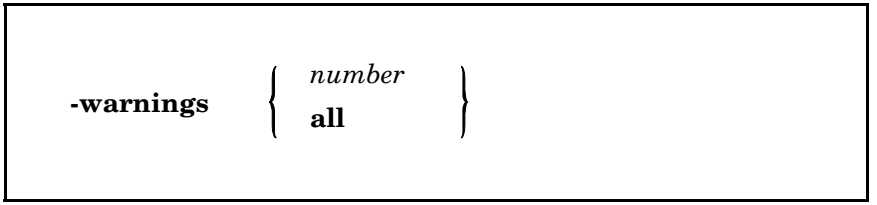
```
-recordlayout  location 14 char, filler 8,
               store_code 4 char, manager_code 4 char,
               emp_count 2 int, filler 34, p_ind 1 char

-r cust_code 2 int, regular_payment 2 uint,
               month_amount_paid 2 uint repeat 12,
               reminder_sent 1 int repeat 12

-recordlayout  cust_code 2 int, regular_payment 2 uint,
               monthly_account
                  {month_amount 2 uint,
                  date_received 6 char,
                  month_paid 1 int,
                  filler 1,
                  reminder
                     {reminder_sent 1 int,
                     date_sent 6 char
                     } repeat 2
                  } repeat 12
```

-warnings

Purpose

To limit the number of warning messages that are issued.

Format



Arguments

number	the maximum number of warning messages to be issued by the application.

Notes

Defaults

FilePort, by default, will issue a maximum of 100 warnings, not including validation warnings.

Warning Suppression

The *number* requested with the **-warnings** option sets an upper limit on warnings encountered while converting records. When the limit specified is exceeded, further warnings are suppressed. When the limit is a number greater than zero and the limit has been reached, a message will indicate

further warnings are being suppressed. This limit does not include warnings that result from validation. Also, this limit does not include messages reporting any problems with severity greater than warning. These warnings and messages are issued regardless of the limit that is specified using the **-warnings** option. Statistics will still be displayed regardless of any limit specified.

Disabling Warning Suppression

When the **all** argument is used, all warning messages are displayed.

Examples

```
-warnings 20
-warnings 0
-warnings all
```

Chapter 5. Mainframe-to-UNIX Examples

The examples presented in this chapter are available online under the *fileport/examples* directory. Each example is held in a file named *mvs_to_unix.exn*, where *n* is the example number. The input files and record layouts are held in the same directory.

Example 1: COBOL Record Layout, Record Format Unchanged

```
fileport -i mvs_input1 -d cobol_layout1
         -o unix_output1
```

Notes:

-i (-infile) specifies the name of the input file as mvs_input1. The record format defaults to fixed, and the length of the input records is picked up from the COBOL record layout as 38 bytes.

-d (-datadictionary) specifies that the record layout exists in a COBOL copybook called cobol_layout1. cobol_layout1 is a standard UNIX text file with contents as follows:

```
01 EMP-RECORD.
   05 FIRST-NAME  PIC X(10).
   05 LAST-NAME   PIC X(10).
   05 PHONE-NO PIC X(12).
   05 EMP-ID   PIC 9(3)COMP-3.
   05 PROJ-ID  PIC 9(5)COMP.
```

-o (-outfile)	specifies the name of the output file as unix_output1. If this file already exists, FilePort terminates with an error message.

The result of the conversion is that the fields FIRST-NAME, LAST-NAME and PHONE-NO are translated from EBCDIC to ASCII; the field EMP-ID is copied unchanged; and, on big-endian systems, the field PROJ-ID is copied unchanged. On little endian systems, such as Intel or Alpha based systems, the bytes within PROJ-ID are reversed.

Example 2: FilePort Record Layout, Record Format Unchanged

```
fileport -infile mvs_input1 fixed 38 \
         -recordlayout  fname 10 char, \
                        lname 10 char, \
                        phone_no 12 char, \
                        emp_id 2 pd, \
                        proj_id 4 uint \
         -outfile unix_output2 overwrite
```

Notes:

-infile specifies the name of the input file as mvs_input1, its record format as fixed, and the length of the input records as 38 bytes.

-recordlayout describes the fields within each record. All field lengths are in bytes.

-outfile specifies the name of the output file as unix_output2, and specifies that if the file already exists, it is to be overwritten.

The result of the conversion is that the fields fname, lname and phone_no are translated from EBCDIC to ASCII; the field emp_id is copied unchanged; and, on big-endian systems, the field proj_id is copied unchanged. On little endian systems, such as Intel or Alpha based systems, the bytes within proj_id are reversed.

Example 3: Conversion to Standard UNIX Text Records

```
fileport -infile mvs_input1 fixed 38 \
         -recordlayout  fname 10 char, \
                        lname 10 char, \
                        phone_no 12 char, \
                        emp_id 2 pd, \
                        proj_id 4 uint \
         -outfile unix_output3 text overwrite
```

Notes:

-infile specifies the name of the input file as mvs_input1, its record format as fixed, and the length of the input records as 38 bytes.

-recordlayout describes the fields within each record. All field lengths are in bytes.

-outfile specifies the name of the output file as unix_output3, and uses the **text** option, which is shorthand for the keywords **linefeed separate compress display noslack**. If unix_output3 already exists, it is to be overwritten.

The result of the conversion is that the fields fname, lname and phone_no are translated from EBCDIC to ASCII; the fields emp_id and proj_id are converted to ASCII display fields. All records are terminated with a linefeed character, all fields are compressed and separated from the previous field by a space.

Example 4: Data Converted to Display, Fixed Length Output Fields

```
fileport -infile mvs_input1 fixed 38 \
         -datadictionary cobol_layout1 \
         -outfile unix_output4 display \
          separate '"~"'
```

Notes:

-infile specifies the name of the input file as mvs_input1, its record format as fixed, and the length of the input records as 38 bytes.

-datadictionary specifies that the record layout exists in a COBOL copybook called cobol_layout1. cobol_layout1 is a standard UNIX text file with contents as follows:

```
01 EMP-RECORD.
   05 FIRST-NAME  PIC X(10).
   05 LAST-NAME   PIC X(10).
   05 PHONE-NO PIC X(12).
   05 EMP-ID   PIC 9(3) COMP-3.
   05 PROJ-ID  PIC 9(5) COMP.
```

-outfile specifies the name of the output file as unix_output4; **display** specifies that binary fields (emp-id, proj-id) are to be converted to displayable characters; **separate "~"** specifies that a tilde is to be inserted between adjacent fields, and causes the record format to default to linefeed terminated. If unix_output4 already exists, FilePort terminates with an error.

The result of the conversion is that the output file is a standard UNIX file type which can be examined with an editor, printed or displayed on the screen. However, since **compress** was not specified, the fields are still of

fixed length. They will therefore appear in columns when displayed, or can be input to, for example, a database load program by specifying their starting and ending positions.

Example 5: Input and Output on Tape, Input Standard IBM Labeled

```
fileport -infile /dev/rmt/0mn label \
'"DSN=WWCABC.SEQ1.TAPE"' \
         -outfile /dev/rmt/1mn linefeed \
                     blocksize 4096
```

Notes:

This example is not available in the *fileport/examples* directory, as it is machine specific.

-infile specifies that the input is a tape with standard IBM labels, mounted on the drive /dev/rmt/0mn. No record format, length or blocksize is required, as the information will be read from the tape headers. The data set name of the file on tape is specified. FilePort will search through the tape(s) until it finds the correct data set.

-outfile specifies that the output is a tape on device /dev/rmt/1mn, the records are to be terminated with a linefeed character, and the block size is 4096 bytes.

No record layout is specified, so it defaults to all character.

The result of the conversion is that every data byte is converted from EBCDIC to ASCII. Each record has a linefeed character appended to it. The output file is written to the output tape device with no label, with a block size of 4096 bytes.

Example 6: Multiple Record Layouts, COBOL Format

```
fileport -infile mvs_input2 fixed 38 \
         -condition \
          emp-record1.country-code eq '"00"' \
         -datadictionary cobol_layout2 us-phone\
         -condition remaining \
         -datadictionary cobol_layout2 \
         -outfile unix_output6 text overwrite
```

Notes:

-infile specifies the name of the input file as mvs_input2, the record format as fixed, and the record length as 38 bytes.

-condition the first occurrence of this option defines the records which have a value "00" in the field emp-record1.country-code. The next occurrence defines all the remaining records.

-datadictionary specifies that the record layout is in a COBOL copybook called cobol_layout2. The contents of cobol_layout2 are as follows:

```
01 EMP-RECORD1.
   05 FIRST-NAME     PIC X(10).
   05 LAST-NAME      PIC X(10).
   05 EMP-ID      PIC 9(3) COMP-3.
   05 COUNTRY-CODE   PIC X(2).
   05 INT-PHONE-NO   PIC X(14).
   05 US-PHONE REDEFINES INT-PHONE-NO.
      10 AREA-CODE   PIC X(3).
      10 PHONE-NO PIC X(7).
      10 STATE-CODE  PIC 9(5) COMP.
```

Each **-datadictionary** option describes the layout for the records that match the preceding condition.

-outfile specifies the name of the output file to be unix_output6, and uses the **text** option which is shorthand for **linefeed separate compress display noslack**. If the output file already exists, it is overwritten.

The result of the conversion is that a standard UNIX text file is created with each record terminated by a linefeed character. The records are translated using different record layouts depending on the value in the field emp-record1.country-code. For all records which have a value "00" in the field emp-record1.country_code, the redefining field US-PHONE is used; for all other records, the default field INT-PHONE-NO is used. All fields are converted to ASCII, are compressed and separated by a space.

Example 7: Multiple Record Layouts, FilePort Format

```
fileport -infile mvs_input2 fixed 38 \
         -condition country_code eq '"00"' \
         -recordlayout  fname 10 char, \
                        lname 10 char, \
                        emp_id 2 pd, \
                        country_code 2 char, \
                        area_code 3 char, \
                        phone_no 7 char, \
                        state_code 4 uint \
         -condition remaining \
         -recordlayout  fname1 10 char, \
                        lname1 10 char, \
                        emp_id1 2 pd, \
                        country_code1 2 char, \
                        int_phone_no 14 char \
         -outfile unix_output7 display \
          separate '","' overwrite
```

Notes:

-infile specifies the name of the input file as mvs_input2, the record format as fixed, and the record length as 38 bytes.

-condition the first occurrence of this option defines the records which have a value "00" in the field country_code. The next occurrence defines all the remaining records.

-recordlayout defines the record layout for the records that match the preceding condition.

-outfile specifies the name of the output file to be unix_output7. All fields are converted to displayable data, separated by ",". If the output file already exists, it is overwritten.

The result of the conversion is that a standard UNIX text file is created with each record terminated by a linefeed character. The records are translated

using different record layouts depending on the value in the field country_code. The first record layout is used for all records which have a value "00" in the field country_code. The second record layout is used for all other records. All data is converted to ASCII, and all fields are separated by ",".

Example 8: Data Piped Directly From ftp Into FilePort

```
{
ftp localhost << MARK1
cd examples
get mvs_input1 -
MARK1
} | \
fileport -infile 0 open fixed 38 \
         -recordlayout  fname 10 char, \
                        lname 10 char, \
                        phone_no 12 char, \
                        emp_id 2 pd, \
                        proj_id 4 uint \
         -outfile unix_output8 overwrite
```

Notes:

ftp connects to the system localhost, transfers the file mvs_input1 from the directory examples, and pipes the transferred file to standard output (the input of the following **fileport** command). The **ftp** command requires that a .netrc file has been set up in the user's home directory to enable automatic login. The .netrc file looks like:

```
machine localhost
login user1
password user1s_passwd
```

Also, the cd and get commands within **ftp** would require editing to provide the appropriate pathname and filename.

-infile specifies that the input is to be taken from standard input (the output of the previous **ftp** command), the record format is fixed, and the record length is 38 bytes.

-recordlayout describes the fields within each record. All field lengths are in bytes.

-outfile specifies the name of the output file as unix_output8, and specifies that if the file already exists, it is to be overwritten.

The data is read directly from the system localhost, converted, and written into a file locally, without any intermediate work files being used. The result of the conversion is that the fields fname, lname and phone_no are translated from EBCDIC to ASCII; the field emp_id is copied unchanged; and, on big-endian systems, the field proj_id is copied unchanged. On little endian systems, such as Intel or Alpha based systems, the bytes within proj_id are reversed.

Example 9: Data Piped Directly From FilePort Into Database Load

```
mknod n_pipe p
{
fileport -infile mvs_input1 fixed 38 \
         -recordlayout  fname 10 char, \
                        lname 10 char, \
                        phone_no 12 char, \
                        emp_id 2 pd, \
                        proj_id 4 uint \
         -outfile n_pipe text separate \
          x'"09"' } | \
bcp table1 in n_pipe -c -Uuser_id -Ppassword

rm n_pipe
```

Notes:

mknod creates a pipe named n_pipe.

-infile specifies the name of the input file as mvs_input1, its record format as fixed, and the length of the input records as 38 bytes.

-recordlayout describes the fields within each record. All field lengths are in bytes.

-outfile specifies that the output is to be written to the named pipe, n_pipe, to be read directly by the following command, records are to be converted to text, with all fields converted to displayable, fields separated by a tab character (hexadecimal 09), and records terminated by a linefeed.

bcp the SYBASE **bcp** utility loads data into a table which was previously created by a command of the form:

```
create table table1
   (first_name varchar(10),
   last_name varchar(10),
   phone_number char(12),
   emp_id smallint not null,
   project_id int not null
   )
go
create index i1 on table1(emp_id)
go
```

The result of the conversion is that the fields fname, lname and phone_no are translated from EBCDIC to ASCII; the fields emp_id and proj_id are converted to displayable, and the resulting records are piped to bcp in the appropriate format (bcp accepts linefeed terminate, tab separated, data) and loaded into the database table table1.

Example 10: Data Piped Directly from FilePort into SyncSort

```
fileport -infile mvs_input2 fixed 38 \
         -condition country-code eq '"00"' \
         -datadictionary cobol_layout2 us-phone \
         -condition remaining \
         -datadictionary cobol_layout2 \
         -outfile 1 open | \
syncsort /infile 0 open fixed 38 /copy \
         /datadictionary cobol_layout2 cobol \
         /condition usa country-code eq '"00"' \
         /outfile unix_output10_1 overwrite stlf\
         /reformat last-name, first-name,
          area-code, phone-no \
         /include usa \
         /outfile unix_output10_2 overwrite stlf\
         /reformat last-name, first-name, \
          int-phone-no \
         /omit usa \
         /end
```

Notes:

-infile specifies the name of the input file as mvs_input2, the record format as fixed, and the record length as 38 bytes.

-condition the first occurrence of this option defines the records which have a value "00" in the field emp-record1.country-code. The next occurrence defines all the remaining records.

-datadictionary specifies that the record layout is in a COBOL copybook called cobol_layout2. The contents of cobol_layout2 are as follows:

```
01 EMP-RECORD1.
   05 FIRST-NAME     PIC X(10).
   05 LAST-NAME      PIC X(10).
   05 EMP-ID      PIC 9(3) COMP-3.
   05 COUNTRY-CODE   PIC X(2).
   05 INT-PHONE-NO   PIC X(14).
   05 US-PHONE REDEFINES INT-PHONE-NO.
      10 AREA-CODE   PIC X(3).
      10 PHONE-NO PIC X(7).
      10 STATE-CODE  PIC 9(5) COMP.
```

Each **-datadictionary** option describes the layout for the records that match the preceding condition.

-outfile specifies that the output is to go to standard out (and therefore into SyncSort as the next step).

syncsort invokes the SyncSort utility to reformat records and split data between two files.

The result of the conversion is that the records are translated using different record layouts depending on the value in the field emp-record1.country-code. For all records which have a value "00" in the field emp-record1.country_code, the redefining field US-PHONE is used; for all other records, the default field INT-PHONE-NO is used. The output records are piped through standard output into the next command, syncsort, which divides the records between two files, depending on the country-code, and reformats the records to keep only the first name, last name and phone number fields.

Part II. Converting from UNIX to Mainframe

Chapter 6. Changing Mainframe-to-UNIX Applications to UNIX-to-Mainframe

FilePort supports file conversions from IBM mainframe to UNIX, as well as from UNIX to IBM mainframe. Each conversion is designed to be the inverse of the other, so if you have a mainframe to UNIX application, you can invert it by following the instructions below, and you will not need to refer to the rest of Part II of this Guide.

Standard Procedure

In general, to invert a mainframe-to-UNIX application, you will need to

- swap your **-infile** and **-outfile** options
- add the keyword **unix** immediately following your new input file specification

For example, if your mainframe-to-UNIX application is

```
fileport -infile mvsfile
         -datadictionary cobol1
         -outfile unixfile text
```

the equivalent UNIX-to-mainframe application will be

```
fileport -infile unixfile unix text
         -datadictionary cobol1
         -outfile mvsfile
```

Exceptions

There are some exceptions to the above procedure.

- The keywords **overwrite** and **append** are only valid on **-outfile** . For example, if your mainframe-to-UNIX application is

```
fileport -in mvsfile
   -data cobol1
   -out unixfile text overwrite
```

the equivalent UNIX-to-mainframe application will be

```
fileport -in unixfile unix text
   -data cobol1
   -out mvsfile overwrite
```

- If you use UNIX pipes (*stdin* or *stdout*) in the application, you will have to switch the file descriptor(s). For example , if your mainframe-to-UNIX application is

```
fileport -in 0 open
   -data cobol1
   -out 1 open
```

the equivalent UNIX-to-mainframe application will be

```
fileport -in 0 open unix
   -data cobol1
   -out 1 open
```

- If you are using labeled tapes as input in your application, you may need to change the **-outfile** specification to provide the blocksize and format information, as well as adding some label options depending on the exact output desired. For example, if your mainframe-to-UNIX application is

```
fileport -in /dev/rmt/0mn labeled
      '"DSNAME=PROD1'.UNIX.FILE1"'
   -data cobol1
   -out unixfile text
```

the equivalent UNIX-to-mainframe application may be

```
fileport -in unixfile unix text
   -data cobol1
   -out /dev/rmt/0mn labeled
'"VOL=SER=002001,DSNAME=PROD1.UNIX.FILE1,
EXPDT=2001/001"'
      blocksize 4096
```

- The **filenumber** argument is valid only on **-infile** .

- The **multivolume** argument is valid only on **-infile** .

Chapter 7. UNIX Input to FilePort

FilePort takes as input a UNIX data file of records to be converted, and one or more record layouts describing the output records.

FilePort uses the output record layout, as well as any special command arguments, to derive the layout of the records in the UNIX file. It identifies the fields in the input file that correspond to fields in the output record layout, and then converts individual fields.

UNIX Data File

FilePort takes a single sequential UNIX file as input. The UNIX input file can be either a binary file or a text file (see under Text Record Format below). The file can be on tape, in the UNIX file system, or piped through standard input. Details of the input are specified using the **-infile** option described in Chapter 9.

Fixed Length Record Format (Default)

FilePort's default input file contains fixed length records, with fixed position fields, with no special field separators or record terminators, so that the records can contain character, decimal and binary data, as would be used by a COBOL program.

If your input file is not of fixed length format, you can use arguments to the **-infile** option to specify one of the alternative record formats below.

Text Record Format

FilePort can take standard UNIX text records as input, where all data is in display format, fields have optionally been compressed by removing spaces and non-significant zeroes, fields are optionally separated by a user-defined special character (comma, space, etc.) and each record is terminated by a linefeed character. Use the **linefeed** argument to the **-infile** option to specify linefeed terminators, the **display** or **coboldisplay** arguments to specify that all input data is in display format, and the **separate** argument to specify field separators. Alternatively, you can specify the **text** argument, which has the same effect as **linefeed display separate noslack**.

Micro Focus Variable Record Format

The input records can be in Micro Focus variable format. Use the **mfvariable** argument to the **-infile** option.

MS-DOS Text Record Format

The input file can consist of MS-DOS text records, such as would be generated on a PC, where each record is terminated by a carriage return and a linefeed. Specify the **crlf** argument to the **-infile** option.

FORTRAN Unformatted Record Format

The input file can contain standard FORTRAN unformatted records with a 4 byte record descriptor before and after each data record. Use the **fortran** argument to the **-infile** option.

Record Layout

The layout of the data records must be described to FilePort in order for it to convert individual data fields. For UNIX to mainframe conversion, the record layout supplied should be the layout required for the mainframe (output) records. FilePort will then derive the layout of the input records from this output record layout and any special command options specified in **-infile** .

The data types expected by FilePort on input are the UNIX equivalents of the mainframe data types. For example, FilePort expects all character data to be in ASCII and translates it from ASCII to EBCDIC. Integer data is expected to be in the same format as mainframe on big-endian UNIX machines and to have its bytes reversed on little-endian machines. Packed decimal data is expected to be in the same format as mainframe. Floating point data is expected to be in IEEE format rather than IBM excess-64 format. Details of the machine representations of the various data types and their conversions are given in Appendix B.

The record layout can be provided through a COBOL copybook (**-datadictionary** option), or through a FilePort **-recordlayout** option. If the records do not all have the same layout, conditions can be specified to determine which layout to use for each record (**-condition** option).

Record Layout in a COBOL Copybook

If the record layout(s) of your output file are available in a separate copybook, FilePort will read the copybook directly when you use the **-datadictionary** option described in Chapter 9. The copybook itself must be available as a standard text file (ASCII text, records terminated by linefeeds) on the UNIX system.

The translation of field values by FilePort is controlled mainly by the USAGE, PICTURE and SIGN clauses in the data description. Each elementary data item, either described explicitly by a data-name or as a filler, is converted. The relationship between COBOL data types and FilePort data types is described in the **-datadictionary** option in Chapter 9 and in Appendix B.

Zoned Decimal and Embedded Sign Data

FilePort assumes that your input file contains Micro Focus equivalent data types. Embedded sign decimal (zoned decimal) data types have different representations in different compilers. If your input contains embedded sign decimal data in an alternative format (used for example by RM COBOL and DEC COBOL), specify the **altzoned** argument on the **-infile** option. Details of the different formats are given in Appendix B.

Alignment Bytes

FilePort expects that any alignment bytes (slack bytes) needed in the mainframe file to align fields because of a COBOL SYNCHRONIZED clause, are also present in the input, and transfers them to the output. If these slack bytes do not exist in the UNIX version of the file, use the **noslack** argument to the **-infile** option. Slack bytes are not expected in the input if you specify **text** as the input record format. See Appendix D for more information on COBOL slack bytes and the behavior of COBOL compilers.

Record Layout in a FilePort Option

When the record layout(s) of your output file are not available as a COBOL data description file, you must define the positions and formats of the fields through the **-recordlayout** option described in Chapter 9.

Multiple Record Layouts

When your file contains records that do not all have identical layouts, use **-datadictionary** and/or **-recordlayout** options to describe all the possible layouts. Then use a **-condition** option for each different layout to define which records have that layout. The **-condition** option defines a set of input data records (e.g., where the **record-type** field has the value **customer**) and specifies which layout to use for records belonging to that set.

Chapter 8. Mainframe Output from FilePort

FilePort interprets fields in your input records according to the record layout you supplied, and generates a sequential output data set with that layout.

Mainframe Data Set

The mainframe data set can be written to IBM standard labeled or unlabeled tape, to a UNIX file system, or piped to standard output. [Mainframe output to IBM standard labeled tape may not be available on all platforms, refer to your release-notes for information on supported platforms]

The default output record format generated by FilePort is fixed (F). See Appendix A for a detailed description of mainframe record formats. Use the **-outfile** option described in Chapter 9 to specify alternative record formats and lengths.

Chapter 9. UNIX-to-Mainframe Command Reference

The fileport Command

You use the **fileport** command to invoke FilePort from the UNIX shell prompt.

The format of the command is:

> **fileport** *option ...*

An *option* consists of a keyword (one of **-condition** , **-datadictionary, -infile** , **-outfile** , **-recordlayout** , **-warning**) plus arguments to the keyword. You can provide the options in any order. The options are described in the following sections.

The following typographical conventions are observed in the format statements of FilePort command options.

- Variables are shown in *italics*. You must substitute an appropriate numerical or text value for a variable.
- Constants are shown in **bold** text and must be specified as shown. You may supply a constant keyword in either lower or upper case and you may abbreviate it (e.g., **-infile, -o, ALTZ, fixed**).
- Braces { } indicate that a choice must be made from the set of items contained in the braces.
- Brackets [ ] indicate that an item is optional.
- An ellipsis ... indicates that the preceding argument or group of arguments may be repeated.

Correction of Options Problems

When FilePort finds problems in the options, you are notified of each problem in turn. Up to four pieces of information are displayed for each problem.

- The line of the option containing the problem.
- An indicator pointing to the position in the line at which the problem was detected.
- A descriptive message preceded by a mnemonic in parentheses.
- If the error was found in the contents of a COBOL data description file, a message indicating which line in the file is in error.

You may use the message mnemonic with the Syncsort Message Utility, syncmsg, to get more information on the exception. See Chapter 11 for a full description of the syncmsg utility.

Completion Status

After you run FilePort, the value of the shell variable **?** indicates the success or otherwise of the conversion.

0 FilePort completed successfully.

100 FilePort completed but output generated may be incorrect or incomplete, e.g. due to the values of the input data not matching the specified data type.

111 FilePort did not complete successfully.

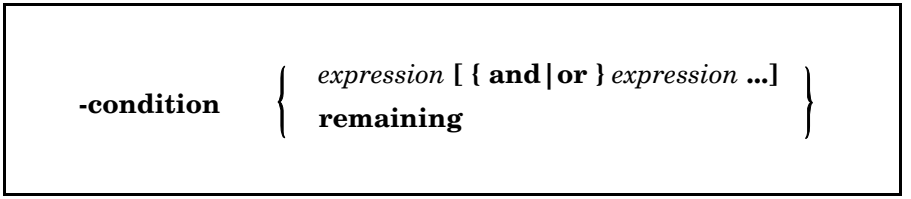
You can check the completion status of FilePort by typing the UNIX command **echo $?** (in Bourne shell or Korn Shell) or **echo $status** (in C-shell) after FilePort has executed.

-condition

Purpose

To define a condition for selecting individual records based on the contents of one or more fields.

Format



where

expression = *field_name* { **eq** | **ne** | **lt** | **gt** | **le** | **ge** } { *field_name* | [*repeat*] *string* | *number* }

For use of parentheses within an expression, see the notes below.

Arguments

field_name the name of a field defined in a **-recordlayout** option or in a COBOL copybook. The field must be uniquely identifiable from all the record layouts. Use qualified field names to distinguish between fields of the same name. Fields can have any datatype in the record layout other than floating point. See Appendix E for information on how to specify qualified field names.

repeat	the number of times the *string* is to be repeated to yield the value used in the expression.
string	a character string constant enclosed in double quotes. To include a double quote in a string, use two double quotes. Since UNIX shells may strip double quotes from the command line, you will generally need to enclose pairs of double quotes within pairs of single quotes. A string can be compared only to a field of type character.
number	a numeric constant, consisting of an optional sign followed by one or more digits with an optional decimal point. A number can be compared only to a numeric field.

Location

Each occurrence of the **-condition** option must be immediately followed by a **-recordlayout** or a **-datadictionary** option. If any conditions are defined, the last **-condition** option must be **-condition remaining** .

Notes

For FilePort to locate the field used in a condition in the input records, one of the following must be true:

The field must occur at the same location (before any variable length arrays) in every input record

or

If the fields in the input are separated, the field must be preceded by the same number of separators in every record.

Each condition is evaluated in sequence for each input record. When a record satisfies a condition, it is assumed to have the layout in the **-datadictionary** option or the **-recordlayout** option immediately following that condition. Further **-condition** options are not checked for that record.

The **remaining** condition is satisfied by all records that do not satisfy any preceding **-condition** option. This must be the last condition option that appears in an application.

You may build compound logical expressions by using **and**s, **or**s and pairs of parentheses. Expressions within parentheses are evaluated first. When the order of evaluation is not indicated by parentheses, **and** is evaluated before **or**. Multiple **and** or **or** operators are evaluated from left to right.

Examples

```
-condition  state-code eq '"CA"' or state-code eq '"NY"'

-condition  (status eq 14 or status eq 15) and
            amount gt 100

-condition  remaining
```

-datadictionary

Purpose

To specify the location of one or more output record layouts held in a COBOL copybook.

Format

-datadictionary *file_name* [*field_name* [,*field_name*...]]

Arguments

file_name the UNIX pathname of a file which contains one or more complete COBOL data descriptions with level 01 entries which define the output record layout(s).

field_name the name of a field defined in the COBOL data descriptions. The field name must be uniquely identifiable. Use qualified field names to distinguish between fields of the same name. See Appendix E for information on how to specify qualified field names.

Location

If you have a single record layout for all your output records, this option may appear anywhere in the options definition. If you have multiple layouts for your output records, each **-datadictionary** option must appear immediately following the corresponding **-condition** option. See the section on the **-condition** option in this chapter for details.

Notes

Defaults

If neither **-datadictionary** nor **-recordlayout** is specified, FilePort treats each record as all character data.

Cobol Language Elements

FilePort supports a data description written to the specifications for Release 4 of *VS COBOL II*. Table 3 on the next page summarizes the various data categories available through *VS COBOL II* and indicates the equivalent FilePort data type for each. Note that not all IBM extensions to the standard *X3.23-1985, American National Standard for Information Systems - Programming Language - COBOL* are supported for translation.

Level 66 (RENAMES), level 77 and level 88 data entries are ignored.

The BLANK WHEN ZERO, EXTERNAL, GLOBAL and VALUE clauses are ignored.

A symbolic character is not supported as a figurative constant.

Multiple Record Layouts

A COBOL copybook may contain multiple record layouts in the form of multiple 01 level data items, or data items qualified by the REDEFINES clause.

When the copybook holds multiple record layouts, you can specify *field_name* argument(s) to provide a unique record layout to FilePort.

COBOL Data Type				FilePort Data Type
Category	**Picture**	**Usage**	**Sign**	
alphabetic	A	[DISPLAY]	none	character
numeric	9 P V	[DISPLAY]	none	decimal, unsigned
numeric	S 9 P V	[DISPLAY]	none	decimal, embedded sign, trailing
numeric	S 9 P V	[DISPLAY]	LEADING	decimal, embedded sign, leading
numeric	S 9 P V	[DISPLAY]	LEADING SEPARATE	decimal, separate sign, leading
numeric	S 9 P V	[DISPLAY]	TRAILING SEPARATE	decimal, separate sign, trailing
numeric	9 P V	BINARY, COMP, COMP-4	none	binary, unsigned integer (see also Appendix B)
numeric	S 9 P V	BINARY, COMP, COMP-4	none	binary, signed integer (see also Appendix B)
numeric	S 9 P V	PACKED-DECIMAL, COMP-3	none	packed decimal
numeric edited	S 9 P V B Z * 0 / , . + - CR DB	[DISPLAY]	none	character
alphanumeric	X	[DISPLAY]	none	character
alphanumeric edited	A X 9 B 0 /	[DISPLAY]	none	character
DBCS	G B	DISPLAY-1	none	**not supported**
external floating point	9 V . E + -	[DISPLAY]	ignored	**not supported**
floating point	none	COMP-1, COMP-2	none	binary, floating point
index	none	INDEX	none	binary, signed integer
pointer	none	POINTER	none	binary, signed integer

Table 3. Data Categories Available Through VS COBOL II

Specify a *field_name* for each data item needed to distinguish between alternative layouts, as follows:

- to choose between complete data descriptions in the copybook, specify the level 01 data item name associated with the desired data description.

- to choose between redefined portions of the data description, specify the name of a data item included in one of the redefinitions.

Designating a data item implicitly designates any group item of which it is a subordinate. Therefore, you cannot specify data items which are subordinate to two group items that redefine each other. For example, you cannot specify two items subordinate to different level 01 items.

When you identify a redefining item of the data description as part of the record layout and it is shorter than the redefined item, the portion of the record not covered by the redefining item is unchanged in the output.

When you do not provide information to distinguish between layouts, FilePort uses the first of any set of choices, as follows:

- when the copybook contains multiple level 01 data items, FilePort chooses the first one.

- when a subordinate data item has a REDEFINES clause, FilePort chooses the redefined item over the redefining item.

A level 66, level 77 or level 88 data item cannot be specified as the *field_name* argument.

When the output record format is fixed, all the record layouts must have the same length, or the record length must be specified in the **-outfile** option.

When the output record length is specified in the **-outfile** option, and the length of any record layout is longer than the specified length, FilePort stops with an error message. If any record layout is shorter than the specified length, the remainder of the record is unchanged in the output.

Examples

```
-datadictionary company.cbl

-datadictionary /general/production/masterfile.cob

-datadictionary copybook_1 level2-BC
```

-infile

Purpose

To define the source of data records to be converted.

Format

-infile *file_identifier* [**unix**] [*record_format*] [*max_length*]
[**blocksize** *block_size*] [**filenumber** *file_number*]
[**multivolume** *num_volumes*] [*field_option*]
[*data_format*]

where

file_identifier = { *file_name* | *file_descriptor* **open** }

record_format = { **crlf** | **fixed** | **fortran** | **mfvariable** | **linefeed** | **text** | **variable** }

field_option = [**separate** [*field_separator*]] [**noslack**]
[**occursdependingon**] [*sequence_name*]

field_separator = { "*a*" | **x**"*xx*" | **o**"*ooo*" }

data_format= { **altzoned** | **coboldisplay** | **display** }

Arguments

file_name	the name of a file which contains records to be converted
file_descriptor	the file descriptor of an open file which contains records to be converted.
max_length	the length in bytes of the records in the file (for fixed records) or the longest record in the file (for other formats). The length for variable records excludes the length of the RDW.
block_size	the block size in bytes for an input file on tape.
file_number	the file number on tape, starting at 1, when you have a tape that contains multiple data files.
num_volumes	the number of tape volumes to be read.
sequence_name	the name of the translation table. See Appendix C for acceptable values for the sequence name argument.

Notes

The input to FilePort is a file on disk, tape, or in standard input, which contains the data you want to convert. Use the **unix** option to indicate that the conversion direction is from UNIX (to mainframe), rather than the default from mainframe to UNIX.

Input Device

Disk. Provide the UNIX file name as the *file_name* argument.

Tape. Provide the tape device file name as the *file_name* argument. The tape should be mounted in the drive before executing FilePort. UNIX tapes are always expected to be unlabeled.

If you have multiple files on the tape and the file you want is not the first one, then you must provide a device name that is both non-rewinding and allows multiple files to be read from tape. Non-rewinding tape device files typically contain an **n** in their name, and device files allowing multiple files have a **b** in their name on some systems, e.g. **/dev/rmt/0mbn**. If in doubt about which device name to use, consult your system administrator.

For tape input, you must provide the block size that was used when writing the tape, through the **blocksize** *block_size* argument. If you have multiple files on the same tape, specify the **filenumber** argument to identify which one you want to convert. If your file spans multiple tape volumes, specify the **multivolume** argument with the number of volumes. [The **multivolume** argument may not be available on all platforms. Refer to your release-notes for information on supported platforms.]

For multivolume input, FilePort expects each volume to be mounted on the same tape drive. At the end of each tape volume, FilePort displays a message on *stderr* and polls the tape drive, waiting for the next tape to be made online. If the input device is an autoloader, the next volume should automatically get mounted at this point. Otherwise, mount the next tape in the drive.

Standard input. Specify the *file_descriptor* as 0, with the **open** keyword and record information. For example, **-infile 0 open fixed 80** .

Record Format and Length

If you do not specify the input record format, the default is **linefeed** if the **separate** field argument is used, and **fixed** otherwise. If you do not provide the maximum record length, FilePort derives it from the output record layout and length. If the specified input record length is shorter than the length derived from the record layout(s), FilePort stops with an error.

Field Options

Field alignment. The input data fields are assumed to be unaligned, i.e. each data field starts in the byte immediately following the last byte of the previous field, unless your record layout is from a COBOL copybook and contains a SYNCHRONIZED clause.

In the case that your record layout is from a COBOL copybook and contains a SYNCHRONIZED clause, FilePort expects to find alignment bytes (slack bytes) in your input records. If these are not present, then specify the **noslack** argument. A detailed description of COBOL slack bytes is given in Appendix D.

Field separators. By default, input fields are expected to be adjacent to one another. If your input fields are separated, for example by a space, a tab, a comma, etc., then use the **separate** argument. If you specify the **separate** argument, FilePort will scan for a field separator between each elementary field in the **-recordlayout** option or the COBOL copybook. If the field separator character in your input is other than the UNIX default (a blank), specify it through the *field_separator* argument. Since the UNIX shells may strip double quotes from the command line, you will generally need to enclose the double quotes around the separator within a pair of single quotes. When you specify the **separate** argument, the record format defaults to **linefeed**.

Occursdependingon. By default, FilePort expands any variable length array defined by an OCCURS DEPENDING ON clause in your copybook to be of the maximum size in the input data. If the array is of variable length in the input, specify the **occursdependingon** argument.

Sequence name. By default, FilePort uses the first ASCII to EBCDIC translation table in Appendix C for UNIX to mainframe data conversion. Alternatively, other sequence names may be specified. Table 1 on page C-1 lists these sequences and gives the page number in the appendix where the associated translation table may be found.

Data Type Conversion

Each UNIX data field is converted to its appropriate mainframe format according to the rules detailed in Appendix B. The format of the input data is further specified by the following three arguments:

Display. If you specify the **display** keyword, FilePort expects the data fields listed in the **datadictionary** (which defines the *output* file format) to

appear in the *input* file in a displayable format. The details of the format are included in Appendix B.

Coboldisplay. If you specify the **coboldisplay** keyword, FilePort expects that the data fields listed in the **datadictionary** (which defines the *output* file format) appear in the *input* file in a displayable format that conforms to the COBOL DISPLAY statement. The details of the format are included in Appendix B. **Coboldisplay** is incompatible with the **compress**, **separate**, and **text** keywords.

Altzoned. If your input records contain display-numeric data with an embedded or overpunched sign (also referred to as zoned decimal data) it may occur in one of two formats. By default, FilePort treats it as Micro Focus format. If your data is in the other format, specify the altzoned keyword. Details of the two formats are shown in Appendix B.

Examples

```
-infile  unix.in unix fixed 100

-infile  unix.to.mainframe unix text 82

-infile  /dev/rmt/0mn fixed 128 blocksize 512 filenumber 2

-infile  0 open linefeed 200 display separate '";"'

-infile /dev/rmt0 unix text blocksize 4096 multivolume 4
```

-outfile

Purpose

To specify the destination of converted records.

Format

-outfile *file_identifier* [**mvs**] [*disposition*] [*record_format*] [*max_length*] [**blocksize** *block_size*] [**spanned**] [**labeled** ["*label_options*"]]

where

file_identifier = { *file_name* | *file_descriptor* **open** }

disposition = { **overwrite** | **append** }

record_format = { **fixed** | **unblockedvariable** | **variable** }

label_options = *label_arg* [**,** *label_arg*...]

label_arg = { **VOL**[**UME**]**=SER=***serial_number* | **DSN**[**AME**]=*dataset_name* | **EXPDT**=*expiry_date* | **RETPD**=*retention_days* | **LABEL**=*protection* }

expiry_date = *yyyy*/*ddd*
yyddd

retention_days = *dddd*

protection = { **PASSWORD** | **NOPWREAD** }

Arguments

file_name	the name of a file which is to receive converted records
file_descriptor	the file descriptor of an open file which is to receive converted records
max_length	the record length in bytes of **fixed** format records, or the maximum record length of **unblockedvariable** or **variable** format records. The length for unblockedvariable or variable records includes the length of the RDW; for example if your longest record contains 100 bytes of data your *max_length* will be 104.
block_size	the block size in bytes for a tape output file, or for a **variable** format disk output file. For a **fixed** output file, the blocksize must be a multiple of the record length. For a **variable** output file, the blocksize must be greater than or equal to *max_length* + 4.
serial_number	a number with one to six digits specifying the volume serial number you want on the first tape. FilePort pads with leading zeros to make the number six digits long. The default is 000001.
dataset_name	name that you want to give to the data set on the tape(s). The data set name can contain letters, digits, $, #, @, and -. The first character cannot be a - or a digit. If the data set name contains any characters other than letters and digits, enclose it within a pair of single quotes. The default value is 'UNIXOUT'.
yyyy	4 digit number specifying the year in which tape expires. This number can not exceed 2999.

yy	2 digit number specifying the year in which the tape expires. Assumed to be prefixed by "19".
ddd	3 digit number between 000 and 366 specifying the day in the year that the tape expires.
dddd	1 to 4 digit specification of number of days from today the tape expires.

Notes

FilePort writes the records after translation to an output file, which can be on disk or tape, or to standard output. Use the **mvs** option to indicate that the conversion direction is (from UNIX) to mainframe, rather than the default from mainframe to UNIX.

Output Device

Disk. FilePort uses the name you provide in the *file_name* argument.

Tape. Provide the tape device file name as the *file_name* argument. The tape must be positioned to the start before executing FilePort.

For multivolume output, FilePort expects each volume to be mounted on the same tape drive. At the end of each tape volume, FilePort displays a message on *stderr* and polls the tape drive, waiting for the next tape to be made online. If the output device is an autoloader, the next volume should automatically get mounted at this point. Otherwise, mount the next tape in the drive. [Multivolume output may not be available on all supported platforms. Refer to your release-notes for information on supported platforms.]

If you want to produce a standard IBM labeled tape, specify the **labeled** argument. [Labeled output may not be available on all supported platforms. Refer to your release-notes for information on supported platforms.]

If your labeled output tape file spans more than one volume, FilePort will generate unique volume serial numbers by incrementing the serial number of the first volume. For labeled output, you can specify the security level by using the **LABEL** argument. The value **PASSWORD** specifies that a valid password will be required to read, write or delete the file, **NOPWREAD** specifies that a password will be required only for write or delete. Refer to

your mainframe documentation for further information on data set passwords. The expiration date can be specified either through **EXPDT** or **RETPD**, but not both. The default is 1 day from today.

Standard output. When you wish FilePort to supply the output records to standard output, specify the file descriptor of standard output and the **open** keyword, plus any record information. For example, **-outfile 1 open** .

File Organization

The converted records are written to a file with sequential organization.

The order of records in the output file is the same as the order of records in the input.

File Disposition

For output to tape, FilePort always overwrites the tape starting at the current position.

For output to a disk file, if the file does not already exist, a new file is created. If you wish to use an existing file, you must specify either **overwrite** or **append**.

overwrite specifies that if the output file already exists, the contents are to be replaced by new output.

append specifies that if the output file already exists, new output is to be appended to the end.

Record Alignment

The converted records are not aligned, i.e. each record starts in the byte immediately following the last byte of the previous record.

Record Format

The default output record format is **fixed**. See Appendix A for a detailed description of the different record formats.

Record Blocking

If the **blocksize** argument is specified, the *block_size* value must follow the rules defined above. If no **blocksize** argument is specified, the default block size used is *max_length* for fixed length records and , and *max_length* + 4 for variable length records.

When deciding on a block size to use, be aware that (a) larger block sizes will generally improve input/output efficiency, (b) larger block sizes will generally increase the amount of data that can be stored on a single tape volume, and (b) some mainframe tape devices may not be able to read blocks shorter than 18 bytes. The largest block size that can be written depends on the particular UNIX operating system and device driver that you are using.

Record Spanning

By default, the output is not spanned. Specify the **spanned** argument to produce spanned output. This argument is required if you have variable length output records that are longer than the block size.

Record Length

FilePort calculates the default output record length for fixed length output records from the length of the record layout specified. If the output records are variable, the default maximum record length is 32756.

Examples

```
-outfile mvs.out mvs

-outfile outfile1 mvs fixed 100 overwrite

-outfile 1 open fixed 100

-outfile /dev/rmt/0mbn blocksize 16384

-outfile /dev/rmt/0mbn variable 136 blocksize 32768

-outfile /dev/rmt0 mvs fixed 100 labeled
'"VOL=SER=100032,DSN=PROD1.SEQ.TAPE1,RETPD=30"'
```

-recordlayout

Purpose

To describe the layout of the mainframe (output) records.

Format

-recordlayout *field* [, *field* ...]

where

field = { *elementary_field* | *composite_field* }

elementary_field = { *fieldname* [*length* [*datatype*] [**repeat** *count*]] | **filler** *length* }

The first format is used to describe a field which is to be converted. The second format is used to allow for portions of the input record which do not need to be converted.

composite_field = *fieldname* **{** *field* [, *field* ...] **}** [**repeat** *count*]

Note that the braces in *composite_field* must appear where shown. They do not indicate a choice of arguments in this case.

Arguments

fieldname the name of a composite or elementary field in the input record layout. The name assigned to a field must adhere to the rules for an identifier defined in Appendix E. The name of a field cannot be the same as the name of an including composite field.

count the number of times a field is repeated consecutively in the input record.

datatype the type of data held in a field. The default data type is character.

length the length of a field, in bytes. The length may be omitted only for a variable length character field at the end of the input record.

Location

This option may appear anywhere in the options definition if all your records have a single record layout. If you have multiple layouts for your input records, each **-recordlayout** option must appear immediately following the corresponding **-condition** option.

Notes

Defaults

If neither **-datadictionary** nor **-recordlayout** is specified, FilePort treats each record as all character data.

Data Types

The **-recordlayout** option allows you to specify your data fields as any of the standard data types as described in Appendix B. If your record contains a field with a non-standard data type, you can specify that part of the record to FilePort as a **filler**. The data will be unaffected by the translation process and will appear in the output file in the same form as in the input file.

Table 4 on the next page summarizes the supported data types and gives the keyword and range of lengths to be used in the **-recordlayout** option for each type.

FilePort Data Type	Keyword	Length (bytes)
binary, floating point	FLOAT	2 -16
binary, signed integer	INTEGER	1 - 256
binary, unsigned integer	UINTEGER	1 - 32,767
character	CHARACTER	1 - 32,767
decimal, edited numeric	EN	1 - 256
decimal, embedded sign	LP, TP	1 - 256
decimal, separate sign	LS, TS	2 - 256
decimal, unsigned	AN	1 - 256
filler	FILLER	1 - 32,767
packed decimal	PD	1 - 256

Table 4. Input Data Types and Lengths Available through ***-recordlayout***

Elementary, Composite and Repeated Fields

The basic building block of the record layout is an *elementary field*, which describes one contiguous portion of data in a record. An elementary field is not subdivided further. In many cases, your record layout will consist simply of a list of elementary fields. When an elementary field in the record layout is repeated several times without any intervening fields or fillers, you can conveniently specify the field only once and add the **repeat** argument to indicate how many times it is repeated.

When two or more consecutive elementary fields in your record layout are repeated several times as a block, it is convenient to consider the repeated elementary fields as a single larger field. A field which is made up of several elementary fields is referred to as a *composite field*. For example, a composite field **full_name** may consist of three elementary fields, **first_name**, **middle_initial** and **last_name**. A composite field can have other composite fields as components, as well as elementary fields.

The elementary fields which make up a composite field do not have to have the same data types. You cannot assign a data type to a composite field.

Multiple Record Layouts

When the output record format is fixed, all the record layouts must have the same length, or the record length must be specified in the **-outfile** option. If any record layout is shorter than the specified length, the remainder of the record is treated as a filler, and is unchanged in the output.

Examples

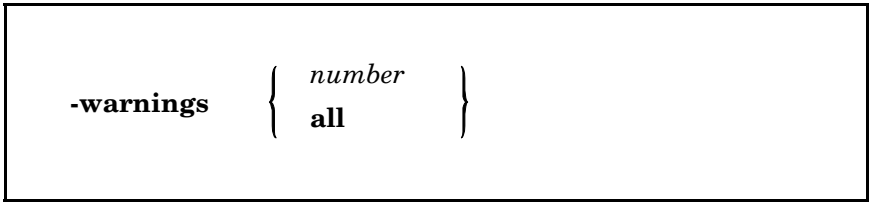
```
-recordlayout  location 14 char, filler 8,
               store_code 4 char, manager_code 4 char,
               emp_count 2 int, filler 34, p_ind 1 char

-r cust_code 2 int, regular_payment 2 uint,
               month_amount_paid 2 uint repeat 12,
               reminder_sent 1 int repeat 12

-recordlayout  cust_code 2 int, regular_payment 2 uint,
               monthly_account
                  {month_amount 2 uint,
                  date_received 6 char,
                  month_paid 1 int,
                  filler 1,
                  reminder
                     {reminder_sent 1 int,
                     date_sent 6 char
                     } repeat 2
                  } repeat 12
```

-warnings

Purpose

To limit the number of warning messages that are issued.

Format



Arguments

number	the maximum number of warning messages to be issued by the application.

Notes

Defaults

FilePort, by default, will issue a maximum of 100 warnings, not including validation warnings.

Warning Suppression

The *number* requested with the **-warnings** option sets an upper limit on warnings encountered while converting records. When the limit specified is exceeded, further warnings are suppressed. When the limit is a number greater than zero and the limit has been reached, a message will indicate

further warnings are being suppressed. This limit does not include warnings that result from validation. Also, this limit does not include messages reporting any problems with severity greater than warning. These warnings and messages are issued regardless of the limit that is specified using the **-warnings** option. Statistics will still be displayed regardless of any limit specified.

Disabling Warning Suppression

When the **all** argument is used, all warning messages are displayed.

Examples

```
-warnings 20
-warnings 0
-warnings all
```

Chapter 10. UNIX-to-Mainframe Examples

The examples presented in this chapter are available online under the *fileport/examples* directory. Each example is held in a file named *unix_to_mvs.ex**n***, where ***n*** is the example number. The input files and record layouts are held in the same directory.

Example 1: COBOL Record Layout, Record Format Unchanged

```
fileport -i unix_input1 unix -d cobol_layout1
         -o mvs_output1
```

Notes:

-i (**-infile**) specifies the name of the input file as unix_input1, and the keyword **unix** to specify unix to mainframe conversion. The record format defaults to fixed, and the length of the input records is calculated from the record layout to be 38 bytes.

-d (**-datadictionary**) specifies that the record layout exists in a COBOL copybook called cobol_layout1. cobol_layout1 is a standard UNIX text file with contents as follows:

```
01 EMP-RECORD.
   05 FIRST-NAME  PIC X(10).
   05 LAST-NAME   PIC X(10).
   05 PHONE-NO PIC X(12).
   05 EMP-ID   PIC 9(3)COMP-3.
   05 PROJ-ID  PIC 9(5)COMP.
```

-o (**-outfile**) specifies the name of the output file as unix_output1. If this file already exists, FilePort terminates with an error message.

In this example, FilePort expects an input file containing fixed length 38 byte records with the specified layout. The result of the conversion is that the fields FIRST-NAME, LAST-NAME and PHONE-NO are translated from ASCII to EBCDIC; the field EMP-ID is copied unchanged; and, on big-endian systems, the field PROJ-ID is copied unchanged. On little endian systems, such as Intel or Alpha based systems, the bytes within PROJ-ID are reversed.

Example 2: FilePort Record Layout, Record Format Unchanged

```
fileport -infile unix_input1 \
         -recordlayout  fname 10 char, \
                        lname 10 char, \
                        phone_no 12 char, \
                        emp_id 2 pd, \
                        proj_id 4 uint \
         -outfile mvs_output2 mvs fixed 38 overwrite
```

Notes:

-infile specifies the name of the input file as unix_input1. The record format defaults to fixed, and the length of the input records is calculated from the record layout to be 38 bytes.

-recordlayout describes the fields within each record. All field lengths are in bytes.

-outfile specifies the name of the output file as mvs_output2; the keyword **mvs** specifies unix to mainframe conversion; records are fixed length of 38 bytes each. If the file already exists, it is to be overwritten.

In this example, FilePort expects an input file containing fixed length 38 byte records with the specified layout. The result of the conversion is that the fields fname, lname and phone_no are translated from ASCII to EBCDIC; the field emp_id is copied unchanged; and, on big-endian systems, the field proj_id is copied unchanged. On little endian systems, such as Intel or Alpha based systems, the bytes within proj_id are reversed.

Example 3: Conversion From Standard UNIX Text Records

```
fileport -infile unix_input2 unix text \
         -recordlayout fname 10 char, \
                       lname 10 char, \
                       phone_no 12 char, \
                       emp_id 2 pd, \
                       proj_id 4 uint \
         -outfile mvs_output3 overwrite
```

Notes:

-infile specifies the name of the input file as unix_input2, and uses the **text** option, which is shorthand for the keywords **linefeed separate display noslack**.

-recordlayout describes the fields within each record of the mainframe file. All field lengths are in bytes.

-outfile specifies the name of the output file as mvs_output3; the record format defaults to fixed, and the record length is calculated from the record layout to be 38 bytes. If mvs_output3 already exists, it is to be overwritten.

In this example, FilePort expects as input a standard UNIX text file where all data is displayable, and all fields are separated by spaces. It converts the fields fname, lname and phone_no from ASCII to EBCDIC, and the fields emp_id and proj_id from ASCII display fields to packed decimal and unsigned integer respectively. All fields are padded or truncated to the length specified in the record layout, and fixed length records are output.

Example 4: Data Converted From Fixed Length Display Fields

```
fileport -infile unix_input3 unix display \
          separate '"~"' \
         -datadictionary cobol_layout1 \
         -outfile mvs_output4 mvs fixed 38
```

Notes:

-infile specifies the name of the input file as unix_input3; the conversion direction is from unix to mainframe; all fields are in displayable format, separated by tildes.

-datadictionary specifies that the record layout exists in a COBOL copybook called cobol_layout1. cobol_layout1 is a standard UNIX text file with contents as follows:

```
01 EMP-RECORD.
   05 FIRST-NAME  PIC X(10).
   05 LAST-NAME   PIC X(10).
   05 PHONE-NO PIC X(12).
   05 EMP-ID   PIC 9(3) COMP-3.
   05 PROJ-ID  PIC 9(5) COMP.
```

-outfile specifies the name of the output file as mvs_output4; the conversion direction is from unix to mainframe; the record format is fixed; and the record length is 38 bytes. If mvs_output4 already exists, FilePort terminates with an error.

In this example, FilePort expects as input a UNIX text file where all data is displayable, and all fields are separated by tildes. It converts the fields fname, lname and phone_no from ASCII to EBCDIC, and the fields emp_id and proj_id from ASCII display fields to packed decimal and unsigned integer respectively. All fields are padded or truncated to the length specified in the record layout, and fixed length records are output.

Example 5: Input and Output on Tape

```
fileport -infile /dev/rmt/1mn unix text \
          blocksize 4096 \
         -outfile /dev/rmt/0mn fixed 128 labeled \
         '"VOL=SER=000256,DSN=/SEQ1.TAPE,RETPD=60"'\
          blocksize 10240
```

Notes:

This example is not available in the *fileport*/*examples* directory, as it is machine specific.

-infile specifies that the input is a tape mounted on the drive /dev/rmt/1mn, and the conversion direction is from UNIX to mainframe. Each input record is terminated by a linefeed character and the maximum input record length defaults to 256 bytes. The tape has a blocksize of 4096 bytes.

-outfile specifies that the output is to be an IBM standard labeled tape on device /dev/rmt/0mn; records are fixed length of 128 bytes each and the block size is 10240 bytes. The tape label will contain the VOLSER and the data set name as specified. Also, the tape expires 60 days from today.

No record layout is specified, so it defaults to all character.

The result of the conversion is that every data byte is converted from ASCII to EBCDIC. Each record has the linefeed character removed and is then truncated or padded to 128 bytes as appropriate and written out to the tape in blocks of 10240 bytes.

Example 6: Multiple Record Layouts, COBOL Format

```
fileport -infile unix_input5 unix text \
         -condition \
          emp-record1.country-code eq '"00"' \
         -datadictionary cobol_layout2 us-phone \
         -condition remaining \
         -datadictionary cobol_layout2 \
         -outfile mvs_output6 overwrite
```

Notes:

-infile specifies the name of the input file as unix_input5; the conversion direction is UNIX to mainframe; the **text** option is shorthand for **linefeed display separate noslack**.

-condition the first occurrence of this option defines the records which have a value "00" in the field emp-record1.country-code. The next occurrence defines all the remaining records.

-datadictionary specifies that the record layout is in a COBOL copybook called cobol_layout2. The contents of cobol_layout2 are as follows:

```
01 EMP-RECORD1.
   05 FIRST-NAME     PIC X(10).
   05 LAST-NAME      PIC X(10).
   05 EMP-ID      PIC 9(3) COMP-3.
   05 COUNTRY-CODE   PIC X(2).
   05 INT-PHONE-NO   PIC X(14).
   05 US-PHONE REDEFINES INT-PHONE-NO.
      10 AREA_CODE   PIC X(3).
      10 PHONE-NO PIC X(7).
      10 STATE-CODE  PIC 9(5) COMP.
```

Each **-datadictionary** option describes the layout for the records that match the preceding condition.

-outfile specifies the name of the output file to be mvs_output6; the record format and length are calculated from the record layout to be fixed length 38 byte records. If the output file already exists, it is to be overwritten.

In this example, FilePort expects as input a standard UNIX text file with variable length fields separated by spaces and records terminated by a linefeed character. The records are translated using different record layouts depending on the value in the fourth field in each record (the field which is to become emp-record1.country-code). For all records which have a value "00" in the fourth field, the redefining field US-PHONE is used; for all other records, the default field INT-PHONE-NO is used. The fields FIRST-NAME, LAST-NAME, COUNTRY-CODE, INT-PHONE-NO, AREA-CODE and PHONE-NO are converted from ASCII to EBCDIC and truncated or padded if necessary to their respective field length. EMP-ID is converted to a two-byte packed decimal field and STATE-CODE is converted to a four-byte unsigned integer field. The records are written out as 38 byte fixed length records.

Example 7: Multiple Record Layouts, FilePort Format

```
fileport -in unix_input4 unix display \
          separate '","' \
         -condition country_code eq '"00"' \
         -recordlayout fname 10 char, \
                       lname 10 char, \
                       emp_id 2 PD, \
                       country_code 2 char, \
                       area_code 3 char, \
                       phone_no 7 char, \
                       state_code 4 uint \
         -condition remaining \
         -recordlayout fname1 10 char, \
                       lname1 10 char, \
                       emp_id1 2 PD, \
                       country_code1 2 char, \
                       int_phone_no 14 char \
         -out mvs_output7 overwrite
```

Notes:

-infile specifies the name of the input file as unix_input4, the conversion direction as UNIX to mainframe; the record format as linefeed terminated, fields containing displayable data and separated by commas.

-condition the first occurrence of this option defines the records which have a value "00" in the field country_code. The next occurrence defines all the remaining records.

-recordlayout defines the record layout for the records that match the preceding condition.

-outfile specifies the name of the output file to be mvs_output7; the record format and length are calculated from the record layout to be fixed length 38 byte records. If the output file already exists, it is to be overwritten.

In this example, FilePort expects as input a standard UNIX text file with fields separated by commas and records terminated by a linefeed character. The records are translated using different record layouts depending on the value in the fourth field in each record (the field which is to become country_code). For all records which have a value "00" in the fourth field, the redefining field us_phone is used; for all other records, the default field int_phone_no is used. The fields first_name, last_name, country_code, int_phone_no, area_code and phone_no are converted from ASCII to EBCDIC. emp_id is converted to a two-byte packed decimal field and state_code is converted to a four-byte unsigned integer field. The records are written out as 38 byte fixed length records.

Example 8: Data Piped Directly From FilePort Into ftp

```
ftp localhost << MARK1
cd fileport
put |"fileport -infile unix_input1 unix \
               -recordlayout fname 10 char, \
                             lname 10 char, \
                             phone_no 12 char, \
                             emp_id 2 pd, \
                             proj_id 4 uint \
               -outfile 1 open " mvs_output8
MARK1
```

Notes:

ftp connects to the system localhost, prepares to transfer data into the file mvs_output8 in directory examples, and then invokes fileport to read the input file, convert the data and pass it through a pipe. The **ftp** command requires that a .netrc file has been set up in the user's home directory to enable automatic login. The .netrc file looks like:

```
machine localhost
login user1
password user1s_passwd
```

Also, the cd and put commands within **ftp** would require editing to provide the appropriate pathname and filename.

-infile specifies that the input file is named unix_input1 and the conversion direction is UNIX to mainframe. The record format defaults to fixed and the record length is calculated from the record layout to be 38 bytes.

-recordlayout describes the fields within each record. All field lengths are in bytes.

-outfile specifies that the output is to be written to standard output.

In this example, FilePort expects a file of fixed length 38 byte records. The result of the conversion is that the fields fname, lname and phone_no are translated from ASCII to EBCDIC; the field emp_id is copied unchanged; and, on big-endian systems, the field proj_id is copied unchanged. On little endian systems, such as Intel or Alpha based systems, the bytes within proj_id are reversed. The records output by FilePort are piped directly into **ftp** and transferred to the system localhost, without any intermediate work files being used.

Example 9: Data Piped Directly From Database Unload Into FilePort

```
mknod n_pipe p

bcp table1 out n_pipe -c -Uuser_id -Ppassword | \
{
fileport -infile n_pipe unix text separate x'""09"'
\
         -recordlayout fname 10 char, \
                       lname 10 char, \
                       phone_no 12 char, \
                       emp_id 2 pd, \
                       proj_id 4 uint \
         -outfile mvs_output9 mvs fixed 38
}
rm n_pipe
```

Notes:

mknod creates a pipe named n_pipe.

bcp the SYBASE **bcp** utility unloads data from a table which was previously created by a command of the form:

```
create table table1
   (first_name varchar(10),
   last_name varchar(10),
   phone_number char(12),
   emp_id smallint not null,
   project_id int not null
   )
go
create index i1 on table1(emp_id)
go
```

-infile specifies the input source as the named pipe n_pipe; the conversion direction as from UNIX; the record format as

UNIX text with fields separated by a tab character (hexadecimal 09).

-recordlayout describes the fields within each record. All field lengths are in bytes.

-outfile specifies that the output is to be written to the file mvs_output9; the conversion direction is to mainframe; the output record format is fixed length 38 byte records.

In this example, the SYBASE **bcp** utility passes unloaded records directly into FilePort without any intermediate file. FilePort expects UNIX text records, where all fields are separated by tabs. It converts the fields fname, lname and phone_no from ASCII to EBCDIC, and truncates or pads them to their respective field length if necessary. It converts emp_id to a two-byte packed decimal field and state_code to a four-byte unsigned integer field. The records are written out as 38 byte fixed length records.

Example 10: Data Piped Directly from SyncSort into FilePort

```
syncsort /infile unix_input1 fixed 38 /copy \
         /datadictionary cobol_layout1 cobol \
         /condition high_id emp-id ge 100 \
         /include high_id \
         /end | \
fileport -infile 0 open unix \
         -datadictionary cobol_layout1 \
         -outfile mvs_output10 mvs fixed 38
```

Notes:

syncsort invokes the SyncSort utility to select a subset of the records for conversion. Only those records with a value greater than or equal to 100 in the emp-id field will be passed through to FilePort.

-infile specifies that the input is to be taken from standard input, which is the output of the preceding command (syncsort). The **unix** keyword specifies that the conversion direction is from UNIX to mainframe.

-datadictionary specifies that the record layout is in a COBOL copybook called cobol_layout1. The contents of cobol_layout1 are as follows:

```
01 EMP-RECORD.
   05 FIRST-NAME  PIC X(10).
   05 LAST-NAME   PIC X(10).
   05 PHONE-NO PIC X(12).
   05 EMP-ID   PIC 9(3) COMP-3.
   05 PROJ-ID  PIC 9(5) COMP.
```

-outfile specifies that the output is to be written to the file mvs_output10; the conversion direction is to mainframe; the output record format is fixed length 38 byte records.

The utility SyncSort is used to select a subset of records to be processed (those records with a value of 100 or greater in the emp-id field), and to pipe these records directly into FilePort. The result of the conversion is that the fields fname, lname and phone_no are translated from ASCII to EBCDIC; the field emp_id is copied unchanged; and, on big-endian systems, the field proj_id is copied unchanged. On little endian systems, such as Intel or Alpha based systems, the bytes within proj_id are reversed.

Appendices

Appendix A. Record Formats

Mainframe Record Formats

Fixed

All records in the file are the same length and do not contain any control information.

Variable

A mainframe variable length record consists of a record descriptor word (RDW) followed by the data. The RDW is 4 bytes long, and contains record length information in standard MVS format. One or more variable length records are grouped together in a block which consists of a block descriptor word (BDW) followed by the records. The BDW is 4 bytes long and contains block length information in standard mainframe format. Optionally, the blocks of data contain segment descriptor words as well. Note that this format always has BDWs, even when there is only one record per block, which is different from the **unblockedvariable** format described below.

Unblockedvariable

An unblockedvariable record consists of a 4-byte record descriptor word (RDW) followed by the data. This format is generated by **ftp** when variable length records are moved from MVS to UNIX using the options BINARY and LOCSITE RDW.

Ftpbinary

An **ftpbinary** record consists of data without any descriptor (RDW). This format is generated by **ftp** when variable length records are moved from MVS to UNIX using the BINARY OPTION.

UNIX Record Formats

Fixed

All records in the file are the same length.

Fortran Unformatted

The file is in standard FORTRAN unformatted format. Each record consists of a 4 byte prefix followed by the data followed by a 4 byte trailer.

Linefeed

The data of each record is followed by a linefeed character. Note that you should not use this format if your records contain binary data, because the values of binary data could then be interpreted as linefeed characters.

Micro Focus Variable (keyword mfvariable)

The file is in standard Micro Focus RECORD SEQUENTIAL variable format. The first data record is preceded by a 128-byte file header. Each record consists of a prefix followed by the data of the record. For a maximum record length of less than 4096 bytes, the prefix is 2 bytes long and for a maximum record length of 4096 bytes or longer, the prefix is 4 bytes long.

MS-DOS (keyword crlf)

The data of each data record is followed by a carriage return and a linefeed character. Note that you should not use this format if your records contain binary data, because the values of binary data could then be interpreted as carriage returns or linefeed characters.

Text

This is shorthand for **linefeed display separate compress noslack**. For more information on **display, separate**, **compress** and **noslack** see the notes on field options and data type conversion in chapters 4 and 9, **-outfile** option.

Variable

Each record has a prefix followed by the data itself. The prefix is a **2 byte** record descriptor word (RDW) which holds the length of the record. The value in the RDW is a big-endian binary number equal to the number of bytes in the data portion of the record, i.e. the value excludes the length of the RDW itself.

Appendix B. Data Types and Conversions

Binary, Floating Point

The floating point data type represents approximations to numeric values, stored in a binary format consisting of a sign, a mantissa (fractional portion) and an exponent (characteristic).

IBM mainframe format

IBM mainframe floating point data is held in excess-64 format. The sign is represented by the first bit; a sign bit of zero indicates a positive number and a sign bit of 1 indicates a negative number. The remaining 7 bits from the first byte contain the characteristic in binary format. The characteristic represents a signed exponent in excess-64 notation. Characteristic values of 0 to 127 indicate exponents of value -64 to +63. The remaining one to fifteen bytes represent the mantissa in binary format. The range of possible values held is approximately 5.4×10^{-79} to $7.2 \times 10^{75.}$

UNIX format

UNIX floating point data is held in IEEE format (*IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1985*). A double precision, IEEE standard, floating point format value has a length of 8 bytes. The range of possible values held is approximately 2.2×10^{-308} to 1.8×10^{308}.

Different UNIX systems use different representations for floating point numbers. Systems based on PA-RISC, SPARC, POWER, and MIPS chips, for example HP-UX, SunOS, AIX, and the MIPS-ABI platforms, use big-endian representation. In big-endian representation, the machine address points to the high-order byte. Systems based on Intel and Alpha chips, for example Digital UNIX, NCR SVR4, Sequent Dynix, and UnixWare, use little-endian representation. In little-endian representation, the machine address points to the low-order byte.

COBOL datadictionary specification

USAGE IS COMP-1 or COMP-2.

FilePort recordlayout specification

FLOAT

Conversion notes

Since the sizes of exponents and mantissas are different in the two formats, conversions between the two may not always be exact. When converting from a mantissa with more bits to a mantissa with less bits precision may be lost, in which case the value in the translated field is rounded up. When converting from an exponent with more bits to an exponent with less bits overflow or underflow may occur, in which case an error message is given.

FilePort uses the storage sequence (big-endian or little-endian) appropriate to the platform on which it runs.

Floating point numbers are not converted to display format. If the **display** option is used on **-outfile** when converting from mainframe to UNIX, a warning message is given and the number is output in IEEE binary format. If the **display** option is used on **-infile** when converting from UNIX to mainframe, the value is treated as an IEEE format binary number.

Binary, Signed Integer (Fixed Point)

The signed integer or fixed point data type represents exact numeric values stored in a binary format in twos complement form. The most significant bit is zero for a positive number and one for a negative number. When described through a COBOL datadictionary the numbers can have an implied decimal point or implied scaling.

IBM mainframe format

IBM mainframes use the big-endian representation for integers. The machine address of an integer points to the high-order byte.

UNIX format

Different UNIX systems use different representations for integers. Systems based on PA-RISC, SPARC, POWER and MIPS chips, for example HP-UX, SunOS, AIX, and the MIPS-ABI platforms, use big-endian representation. In big-endian representation the machine address points to the high-order byte. Systems based on Intel and Alpha chips, for example Digital UNIX,

NCR SVR4, Sequent Dynix, UnixWare, use little-endian representation. In little-endian representation the machine address points to the low-order byte.

COBOL datadictionary specification

USAGE IS BINARY, COMP, or COMP-4.

FilePort recordlayout specification

INTEGER

Conversion notes

FilePort uses the storage sequence (big-endian or little-endian) appropriate to the platform on which it runs.

When the **display** keyword is used to convert unsigned binary data, the length of the display field is the sum of the following:

- the minimum length necessary to hold the largest value that the integer fields can hold
- one byte for the sign
- one byte for the explicit decimal point where the signed integer has an implicit decimal point (COBOL PIC V or PIC P9)
- the appropriate number of bytes where the signed integer has implicit scaling (COBOL PIC P)

For example, a two-byte signed integer field which has neither implicit decimal point nor scaling specified, such as one specified by PIC S9(4) BINARY, will expand to a six character display field: the sign plus five digits (since the largest decimal number which can be stored in a two byte binary field is X'FFFF" or decimal 65535, which requires five decimal digits).

When the **coboldisplay** keyword and the **-datadictionary** option are used to convert signed integers, the length of the display field is one for the sign plus the number of digits indicated by the PIC clause of the field in the COBOL copybook. For example, a field defined as PIC S9(4) BINARY would be converted to a five character display: the sign plus four digits.

Implicit decimal points and implicit scaling are ignored when the **coboldisplay** keyword is used.

When signed integers are converted to displayable format using the **coboldisplay** keyword and the **-recordlayout** option, the length of the output field is the same as that produced when the **display** keyword is used with **recordlayou**t.

The fields converted using the **display** and **coboldisplay** options are padded by default with leading zeroes when needed. The fields converted using the combination of the **display** and **padspace** options are padded with the leading spaces instead.

The placement of the sign for coboldisplay data conforms to that produced by the COBOL DISPLAY statement. For display data, the Edited Numeric data type determines sign placement.

Binary, Unsigned

The unsigned integer data type represents exact positive numeric values stored in binary format. The most significant bit is interpreted as part of the numeric value, not as a sign. Other rules of format and conversion are the same as for the Binary, Signed Integer format, except that when converting an unsigned integer to display format, the length of the display field is one less than when converting the equivalent signed integer to display format.

Character

The character data type represents displayable data and control characters each occupying one byte of storage.

IBM mainframe format

Each byte value is interpreted as an EBCDIC character according to the table in Appendix C.

UNIX format

Each byte value is interpreted as an ASCII character according to the table in Appendix C.

COBOL datadictionary specification

The PICTURE clause contains an A, an X, or an editing character, USAGE IS DISPLAY.

FilePort recordlayout specification

CHARACTER

Conversion notes

FilePort translates each character according to the appropriate table in Appendix C, that is EBCDIC to ASCII for mainframe to UNIX and ASCII to EBCDIC for UNIX to mainframe.

Decimal, Edited Numeric

The edited numeric data type represents displayable numeric digits with leading zeroes suppressed, and a sign immediately preceding the leftmost digit. On UNIX systems the data type can also contain an explicit decimal point.

IBM mainframe format

The IBM mainframe data can consist of optional leading spaces, followed by an optional sign character, followed by decimal digits. Each space, sign and decimal digit takes up one byte and is represented by its 8-bit EBCDIC code. Any characters to the left of the sign character are ignored. An EBCDIC minus sign (X"60") represents a negative number. Any other character (or no sign character) represents a positive number.

UNIX format

The UNIX data can consist of optional leading spaces, followed by an optional minus sign, followed by zero or more decimal digits with optional decimal point and thousand separators as specified by the current locale.

COBOL datadictionary specification

none

FilePort recordlayout specification

EN

Conversion notes

FilePort translates each digit-only byte according to the appropriate table in Appendix C. If neither **display** nor **coboldisplay** is specified, it converts the byte containing the sign according to Table 5 on the next page, using the appropriate UNIX format depending on the presence or absence of the **altzoned** keyword on the UNIX file option.

If the **display** keyword is specified on the **-outfile** option with an embedded-sign decimal, the length of the output field is the sum of the following:

- the number of digits (including the signed one)
- one byte for the sign,
- one byte for the explicit decimal point where the embedded-sign decimal has an implicit decimal point (COBOL PIC V or PIC P9)
- the appropriate number of bytes where the embedded-sign decimal has implicit scaling (COBOL PIC P)

For example, a four-byte decimal field with trailing embedded sign and an implied decimal point, such as one specified by PIC S99V99 SIGN IS TRAILING, will expand to a six character **display** field: the sign plus four digits plus the decimal point.

If the **coboldisplay** keyword is specified on the **-outfile** option with an embedded-sign decimal, input, the length of the output field is the number of digits bytes plus one byte for the sign. Implicit scaling and implicit decimal points are ignored.

For example, a four-byte decimal field with trailing embedded sign and an implied decimal point, such as one specified by PIC S99V99 SIGN IS

TRAILING, will expand to a five character **coboldisplay** field: the sign plus four digits.

The fields converted using the **display** and **coboldisplay** options are padded by default with leading zeroes when needed. The fields converted using the combination of the **display** and **padspace** options are padded with the leading spaces instead.

If the **display** keyword is specified, the sign follows the formatting specifications for Decimal, Edited Numeric. If the **coboldisplay** keyword is specified, the placement of the sign mimics the action of the COBOL DISPLAY statement for the copybook specified by the **-datadictionary** keyword. If **coboldisplay** is specified with the **-recordlayout** option, a trailing plus sign (X'2B') is used for positive numbers and a trailing minus sign (X'2D') is used for negative numbers.

Invalid embedded sign decimals are converted as character data.

Decimal, Embedded Sign

The embedded sign data type, also known as overpunched sign or zoned decimal, is made up of decimal digits with a sign embedded in the first or last byte. The sign replaces the first four bits of the byte holding the most significant digit (leading embedded sign) or the least significant digit (trailing embedded sign).

IBM mainframe format

Each digit except the leading or trailing one occupies a single byte and is represented by its 8-bit EBCDIC code. The leading or trailing digit has the digit value in the right hand half of the byte and in the left hand half of the byte has either (hexadecimal) A, C (preferred), E or F to indicate a positive number or (hexadecimal) B or D (preferred) to indicate a negative number.

UNIX format

Each digit except the leading or trailing one occupies a single byte and is represented by its 8-bit ASCII code. The leading or trailing digit has one of two formats, depending whether the default (Micro Focus) format is used or the alternative format is specified through use of the **altzoned** argument. The two formats are shown in Table 5 below.

COBOL datadictionary specification

The PICTURE clause contains only numeric forms (S, 9, V, P), a sign is specified, USAGE IS DISPLAY.

FilePort recordlayout specification

LP (leading embedded sign), TP (trailing embedded sign).

Conversion notes

FilePort translates each digit-only byte according to the appropriate table in Appendix C. If neither **display** nor **coboldisplay** is specified, it converts the byte containing the sign according to Table 5 on the next page, using the appropriate UNIX format depending on the presence or absence of the **altzoned** keyword on the UNIX file option.

If the **display** keyword is specified on the **-outfile** option with an embedded-sign decimal, the length of the output field is the sum of the following:

- the number of digits (including the signed one)
- one byte for the sign
- one byte for the explicit decimal point where the embedded-sign decimal has an implicit decimal point (COBOL PIC V or PIC P9)
- the appropriate number of bytes where the embedded-sign decimal has implicit scaling (COBOL PIC P)

For example, a four-byte decimal field with trailing embedded sign and an implied decimal point, such as one specified by PIC S99V99 SIGN IS TRAILING, will expand to a six character **display** field: the sign plus four digits plus the decimal point.

If the **coboldisplay** keyword is specified on the **- outfile** option with an embedded-sign decimal, input, the length of the output field is the number of digits bytes plus one byte for the sign. Implicit scaling and implicit decimal points are ignored.

For example, a four-byte decimal field with trailing embedded sign and an implied decimal point, such as one specified by PIC S99V99 SIGN IS TRAILING, will expand to a five character **coboldisplay** field: the sign plus four digits.

The fields converted using the **display** and **coboldisplay** options are padded by default with leading zeroes when needed. The fields converted using the combination of the **display** and **padspace** options are padded with the leading spaces instead.

Digit-plus-sign	IBM Mainframe Values		UNIX Values			
			Default (Micro Focus)		Altzoned	
	Hex	EBCDIC character	Hex	ASCII character	Hex	ASCII character
+0	C0	{	30	0	7B	{
+1	C1	A	31	1	41	A
+2	C2	B	32	2	42	B
+3	C3	C	33	3	43	C
+4	C4	D	34	4	44	D
+5	C5	E	35	5	45	E
+6	C6	F	36	6	46	F
+7	C7	G	37	7	47	G
+8	C8	H	38	8	48	H
+9	C9	I	39	9	49	I
-0	D0	}	70	p	7D	}
-1	D1	J	71	q	4A	J
-2	D2	K	72	r	4B	K
-3	D3	L	73	s	4C	L
-4	D4	M	74	t	4D	M
-5	D5	N	75	u	4E	N
-6	D6	O	76	v	4F	O
-7	D7	P	77	w	50	P
-8	D8	Q	78	x	51	Q
-9	D9	R	79	y	52	R

Table 5. Formats of Digit with Embedded Sign

If the **display** keyword is specified, the sign follows the formatting specifications for Decimal, Edited Numeric. If the **coboldisplay** keyword is specified, the placement of the sign mimics the action of the COBOL DISPLAY statement for the copybook specified by the **-datadictionary** keyword. If **coboldisplay** is specified with the **-recordlayout** option, a trailing plus sign (X'2B') is used for positive numbers and a trailing minus sign (X'2D') is used for negative numbers.

When invalid embedded sign data is found in the input field (a digit position containing other than the appropriate EBCDIC or ASCII digit value, or the digit with embedded sign position containing other than one of the values in Table 5), an error message is output, and the entire field is converted as character data.

Decimal, Separate Sign

Each field is made up of decimal digits, with a separate sign following the least significant digit (trailing separate sign) or preceding the most significant digit (leading separate sign).

IBM mainframe format

Each digit occupies a single byte and is represented by its 8-bit EBCDIC code. The recognized sign characters are EBCDIC plus (X"4E") for positive numbers and EBCDIC minus (X"60") for negative numbers.

UNIX format

Each digit occupies a single byte and is represented by its 8-bit ASCII code. The recognized sign characters are ASCII plus (X"2B") for positive numbers and ASCII minus (X"2D") for negative numbers.

COBOL datadictionary specification

The PICTURE clause contains only numeric forms (S, 9, V, P), SIGN IS SEPARATE, USAGE IS DISPLAY.

FilePort recordlayout specification

LS (leading separate sign), TS (trailing separate sign)

Conversion notes

FilePort translates each character according to the appropriate table in Appendix C, that is EBCDIC to ASCII for mainframe to UNIX and ASCII to EBCDIC for UNIX to mainframe.

When invalid separate sign data is found in the input field (a digit or sign position containing other than the appropriate EBCDIC or ASCII digit or sign value), an error message is output, and the entire field is converted as character data.

When the **display** keyword is specified on the **-outfile** option with separate sign input, the length of the display field is the sum of the following:

- the length of the separate sign field
- one byte for the explicit decimal point where the separate sign data has an implicit decimal point (COBOL PIC V or PIC P)
- appropriate number of bytes where the separate sign data has implicit scaling (COBOL PIC P).

When the **coboldisplay** keyword is specified on the **-outfile** option with separate sign input, the length of the display field is the length of the separate sign field. Implicit decimal points and implicit scaling are ignored.

The fields converted using the **display** and **coboldisplay** options are padded by default with leading zeroes when needed. The fields converted using the combination of the **display** and **padspace** options are padded with the leading spaces instead.

If the **display** keyword is specified, the sign follows the formatting specifications for Decimal, Edited Numeric. If the **coboldisplay** keyword is specified, the placement of the sign mimics the action of the COBOL DISPLAY statement for the copybook specified by the **datadictionary** keyword. If **coboldisplay** is specified with the **recordlayout** option, a trailing plus sign (X'2B') is used for positive numbers and a trailing minus sign (X'2D') is used for negative numbers.

Invalid separate-sign decimals are converted as character data.

Decimal, Unsigned

Unsigned decimal is a numeric data type made up of decimal digits.

IBM mainframe format

Each decimal digit occupies one byte and is represented by its 8-bit EBCDIC code.

UNIX format

Each decimal digit occupies one byte and is represented by its 8-bit ASCII code.

COBOL datadictionary specification

The PICTURE clause contains only numeric forms (9, V, P), USAGE IS DISPLAY.

FilePort recordlayout specification

AN

Conversion notes

FilePort translates each character according to the appropriate table in Appendix C, that is EBCDIC to ASCII for mainframe to UNIX and ASCII to EBCDIC for UNIX to mainframe.

When invalid unsigned decimal data is found in the input field (a byte containing other than the appropriate EBCDIC or ASCII digit value), an error message is output, and the entire field is converted as character data.

When the **display** keyword is specified on the **-outfile** option with unsigned decimal input, the length of the display fields is the sum of the following:

- the length of the unsigned decimal field

- one byte for the explicit decimal point where the unsigned decimal data has an implicit decimal point (COBOL PIC V or PIC P)

- the appropriate number of bytes where the unsigned decimal data has implicit scaling (COBOL PIC P).

When the **coboldisplay** keyword is specified on the **outfile** option with unsigned decimal input, the length of the display field is the length of the unsigned decimal field. Implicit decimal points and implicit scaling are ignored.

The fields converted using the **display** and **coboldisplay** options are padded by default with leading zeroes when needed. The fields converted using the combination of the **display** and **padspace** options are padded with the leading spaces instead.

Invalid unsigned decimals are converted as character data.

Packed Decimal

Packed decimal is a numeric data type consisting of decimal digits and a sign.

IBM mainframe format

Each digit is represented by its 4-bit binary equivalent. Each byte contains two digits, except the last byte which contains the least significant digit in the first four bits, and the sign in the last 4 bits. The following values in the last 4 bits indicate a positive number: (hexadecimal) A, C (preferred), E, F. The following indicate a negative number: (hexadecimal) B, D (preferred).

UNIX format

The UNIX format for packed decimal is the same as the IBM mainframe format.

COBOL datadictionary specification

USAGE IS COMP-3 or PACKED-DECIMAL.

FilePort recordlayout specification

PD

Conversion notes

When invalid packed decimal data is found in the input field (a digit position containing other than hexadecimal 0 through 9, or sign position containing other than hexadecimal A through F), an error message is output, and the entire field is converted as character data.

When packed decimal data is converted using the **display** keyword, the length of the display field is the sum of the following:

- minimum length necessary to hold the largest value that the packed decimal field can hold
- one byte for the sign
- one byte for the explicit decimal point where the packed decimal field has an implicit decimal point (COBOL PIC V or PICP9)
- the appropriate number of bytes where the packed decimal field has implicit scaling (COBOL PIC P).

For example, a two byte packed decimal field specified as PIC S9V99 COMP-3 would be converted to a five character display field: three digits plus the sign plus the decimal point.

When packed decimal data is converted using the **coboldisplay** keyword and a **-datadictionary** option, the length of the display field is the sum of the following:

- the number of digits indicated by the PIC clause of the field in the COBOL copybook
- one for the sign

Implicit decimal points and implicit scaling are ignored. For example, a two byte packed decimal field specified as PIC S9V99 COMP-3 would be converted to a four character **coboldisplay** field: three digits plus the sign.

The fields converted using the **display** and **coboldisplay** options are padded by default with leading zeroes when needed. The fields converted using the combination of the **display** and **padspace** options are padded with the leading spaces instead.

Invalid packed decimal fields are converted as character data.

Appendix C. Character Translation Tables

Alternative Character Sequences

The first column in the following table lists the alternative sequence names that may be specified to FilePort. The second column provides the corresponding code page number for the translation table as defined in the IBM document *Character Data Representation Architecture Level 2 Registry, Second Edition* (December 1993), SC09-1391-01. The third and fourth columns list the starting page numbers in this appendix for the EBCDIC to ISO 8859-1 and ISO 8859-1 to EBCDIC translation tables, respectively, for each sequence. IBM translation table Code Page 0819 - ISO/ANSI Multilingual specifies the ISO 8859-1 character sequence.

Sequence Name	IBM Code Page	Translation Table Page Number	
		EBCDIC to ISO	ISO to EBCDIC
CP37USA	37	C-12	C-62
CP297FRANCE	297	C-17	C-67
CP500GLOBAL	500	C-22	C-72
CP273GERMANY	273	C-27	C-77
CP277DENMARK	277	C-32	C-82
CP278SWEDEN	278	C-37	C-87
CP280ITALY	280	C-42	C-92
CP284SPAIN	284	C-47	C-97
CP285UK	285	C-52	C-102
CP871ICELAND	871	C-57	C-107

Table 1. Alternative Character Sequences Supported by FilePort

EBCDIC to ASCII

The following table shows the 256 EBCDIC character codes and the ASCII character codes which result from the EBCDIC to ASCII translation. This is the default table used when mainframe data is translated to UNIX.

EBCDIC Character Data		ASCII Translation	
Control/ Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
PF/SEL	04	ST	9C
HT	05	HT	09
LC/RNL	06	SSA	86
DEL	07	DEL	7F
GE	08	EPA	97
SPS	09	RI	8D
SMM/RPT	0A	SS2	8E
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO	0E	SO/LS1	0E
SI	0F	SI/LS0	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
TM/DC3	13	DC3	13
RES	14	OSC	9D
NL	15	NEL	85
BS	16	BS	08
IL/POC	17	ESA	87
CAN	18	CAN	18
EM	19	EM	19
CC/UBS	1A	PU2	92

EBCDIC Character Data		ASCII Translation	
Control/ Graphic	Hex Value	Character	Hex Value
CU1	1B	SS3	8F
IFS	1C	FS	1C
IGS	1D	GS	1D
IRS	1E	RS	1E
IUS/ITB	1F	US	1F
DS	20		80
SOS	21		81
FS	22		82
WUS	23		83
BYP	24	IND	84
LF	25	LF	0A
ETB	26	ETB	17
ESC	27	ESC	1B
SA	28	HTS	88
SFE	29	HTJ	89
SM	2A	VTS	8A
CU2/CSP	2B	PLD	8B
MFA	2C	PLU	8C
ENQ	2D	ENQ	05
ACK	2E	ACK	06
BEL	2F	BEL	07
	30	DCS	90
	31	PU1	91
SYN	32	SYN	16
IR	33	STS	93
PN/PP	34	CCH	94
RS/TRN	35	MW	95

EBCDIC Character Data		ASCII Translation	
Control/ Graphic	Hex Value	Character	Hex Value
UC/NBS	36	SPA	96
EOT	37	EOT	04
SBS	38		98
IT	39		99
RFF	3A		9A
CU3	3B	CSI	9B
DC4	3C	DC4	14
NAK	3D	NAK	15
	3E	PM	9E
SUB	3F	SUB	1A
Sp	40	Sp	20
	41	NBSP	A0
	42	¡	A1
	43	¢	A2
	44	£	A3
	45	¤	A4
	46	¥	A5
	47	|	A6
	48	§	A7
	49		A8
¢	4A	Õ	D5
.	4B	.	2E
<	4C	<	3C
(	4D	(	28
+	4E	+	2B
|	4F	|	7C
&	50	&	26
	51	©	A9
	52	ª	AA
	53	«	AB
	54	¬	AC
	55	-(SHY)	AD
	56	®	AE

EBCDIC Character Data		ASCII Translation	
Control/ Graphic	Hex Value	Character	Hex Value
	57	_	AF
	58	°	B0
	59	±	B1
!	5A	!	21
$	5B	$	24
*	5C	*	2A
)	5D	)	29
;	5E	;	3B
¬	5F	~	7E
-	60	-	2D
/	61	/	2F
	62	²	B2
	63	³	B3
	64	´	B4
	65	µ	B5
	66	¶	B6
	67	·	B7
	68	¸	B8
	69	¹	B9
|	6A	Ë	CB
,	6B	,	2C
%	6C	%	25
_	6D	_	5F
>	6E	>	3E
?	6F	?	3F
	70	º	BA
	71	»	BB
	72	¼	BC
	73	½	BD
	74	¾	BE
	75	¿	BF
	76	À	C0
	77	Á	C1
	78	Â	C2

EBCDIC Character Data		ASCII Translation	
Control/ Graphic	Hex Value	Character	Hex Value
`	79	`	60
:	7A	:	3A
#	7B	#	23
@	7C	@	40
'	7D	'	27
=	7E	=	3D
"	7F	"	22
	80	Ã	C3
a	81	a	61
b	82	b	62
c	83	c	63
d	84	d	64
e	85	e	65
f	86	f	66
g	87	g	67
h	88	h	68
i	89	i	69
	8A	Ä	C4
	8B	Å	C5
	8C	Æ	C6
	8D	Ç	C7
	8E	È	C8
	8F	É	C9
	90	Ê	CA
j	91	j	6A
k	92	k	6B
l	93	l	6C
m	94	m	6D
n	95	n	6E
o	96	o	6F
p	97	p	70
q	98	q	71
r	99	r	72
	9A	^	5E

EBCDIC Character Data		ASCII Translation	
Control/ Graphic	Hex Value	Character	Hex Value
	9B	Ì	CC
	9C	Í	CD
	9D	Î	CE
	9E	Ï	CF
	9F	Ð	D0
	A0	Ñ	D1
	A1	å	E5
s	A2	s	73
t	A3	t	74
u	A4	u	75
v	A5	v	76
w	A6	w	77
x	A7	x	78
y	A8	y	79
z	A9	z	7A
	AA	Ò	D2
	AB	Ó	D3
	AC	Ô	D4
	AD	[	5B
	AE	Ö	D6
	AF		D7
	B0	Ø	D8
	B1	Ù	D9
	B2	Ú	DA
	B3	Û	DB
	B4	Ü	DC
	B5	Ý	DD
	B6	Þ	DE
	B7	ß	DF
	B8	à	E0
	B9	á	E1
	BA	â	E2
	BB	ã	E3
	BC	ä	E4

EBCDIC Character Data		ASCII Translation	
Control/ Graphic	Hex Value	Character	Hex Value
	BD	]	5D
	BE	æ	E6
	BF	ç	E7
{	C0	{	7B
A	C1	A	41
B	C2	B	42
C	C3	C	43
D	C4	D	44
E	C5	E	45
F	C6	F	46
G	C7	G	47
H	C8	H	48
I	C9	I	49
	CA	è	E8
	CB	é	E9
	CC	ê	EA
	CD	ë	EB
	CE	ì	EC
	CF	í	ED
}	D0	}	7D
J	D1	J	4A
K	D2	K	4B
L	D3	L	4C
M	D4	M	4D
N	D5	N	4E
O	D6	O	4F
P	D7	P	50
Q	D8	Q	51
R	D9	R	52
	DA	î	EE
	DB	ï	EF
	DC	ð	F0
	DD	ñ	F1

EBCDIC Character Data		ASCII Translation	
Control/ Graphic	Hex Value	Character	Hex Value
	DE	ò	F2
	DF	ó	F3
\	E0	\	5C
	E1	APC	9F
S	E2	S	53
T	E3	T	54
U	E4	U	55
V	E5	V	56
W	E6	W	57
X	E7	X	58
Y	E8	Y	59
Z	E9	Z	5A
	EA	ô	F4
	EB	õ	F5
	EC	ö	F6
	ED		F7
	EE	ø	F8
	EF	ù	F9
0	F0	0	30
1	F1	1	31
2	F2	2	32
3	F3	3	33
4	F4	4	34
5	F5	5	35
6	F6	6	36
7	F7	7	37
8	F8	8	38
9	F9	9	39
	FA	ú	FA
	FB	û	FB
	FC	ü	FC
	FD	ý	FD
	FE	þ	FE

ASCII to EBCDIC

The following table shows the 256 ASCII character codes and the EBCDIC character codes which result from the ASCII to EBCDIC translation. This is the default table used when UNIX data is translated to mainframe.

ASCII Character Data		EBCDIC Translation	
Character	**Hex Value**	**Control/ Graphic**	**Hex Value**
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
EOT	04	EOT	37
ENQ	05	ENQ	2D
ACK	06	ACK	2E
BEL	07	BEL	2F
BS	08	BS	16
HT	09	HT	05
LF	0A	LF	25
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO/LS1	0E	SO	0E
SI/LS0	0F	SI	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	TM/DC3	13
DC4	14	DC4	3C
NAK	15	NAK	3D
SYN	16	SYN	32
ETB	17	ETB	26
CAN	18	CAN	18
EM	19	EM	19

ASCII Character Data		EBCDIC Translation	
Character	**Hex Value**	**Control/ Graphic**	**Hex Value**
SUB	1A	SUB	3F
ESC	1B	ESC	27
FS	1C	IFS	1C
GS	1D	IGS	1D
RS	1E	IRS	1E
US	1F	IUS/ITB	1F
Sp	20	Sp	40
!	21	!	5A
"	22	"	7F
#	23	#	7B
$	24	$	5B
%	25	%	6C
&	26	&	50
'	27	'	7D
(	28	(	4D
)	29	)	5D
*	2A	*	5C
+	2B	+	4E
,	2C	,	6B
-	2D	-	60
.	2E	.	4B
/	2F	/	61
0	30	0	F0
1	31	1	F1
2	32	2	F2
3	33	3	F3

ASCII Character Data		EBCDIC Translation	
Character	Hex Value	Control/ Graphic	Hex Value
4	34	4	F4
5	35	5	F5
6	36	6	F6
7	37	7	F7
8	38	8	F8
9	39	9	F9
:	3A	:	7A
;	3B	;	5E
<	3C	<	4C
=	3D	=	7E
>	3E	>	6E
?	3F	?	6F
@	40	@	7C
A	41	A	C1
B	42	B	C2
C	43	C	C3
D	44	D	C4
E	45	E	C5
F	46	F	C6
G	47	G	C7
H	48	H	C8
I	49	I	C9
J	4A	J	D1
K	4B	K	D2
L	4C	L	D3
M	4D	M	D4
N	4E	N	D5
O	4F	O	D6
P	50	P	D7
Q	51	Q	D8
R	52	R	D9
S	53	S	E2
T	54	T	E3

ASCII Character Data		EBCDIC Translation	
Character	Hex Value	Control/ Graphic	Hex Value
U	55	U	E4
V	56	V	E5
W	57	W	E6
X	58	X	E7
Y	59	Y	E8
Z	5A	Z	E9
[	5B		AD
\	5C	\	E0
]	5D		BD
^	5E		9A
_	5F	_	6D
`	60	`	79
a	61	a	81
b	62	b	82
c	63	c	83
d	64	d	84
e	65	e	85
f	66	f	86
g	67	g	87
h	68	h	88
i	69	i	89
j	6A	j	91
k	6B	k	92
l	6C	l	93
m	6D	m	94
n	6E	n	95
o	6F	o	96
p	70	p	97
q	71	q	98
r	72	r	99
s	73	s	A2
t	74	t	A3
u	75	u	A4
v	76	v	A5

ASCII Character Data		EBCDIC Translation	
Character	Hex Value	Control/ Graphic	Hex Value
w	77	w	A6
x	78	x	A7
y	79	y	A8
z	7A	z	A9
{	7B	{	C0
|	7C	|	4F
}	7D	}	D0
~	7E	¬	5F
DEL	7F	DEL	07
	80	DS	20
	81	SOS	21
	82	FS	22
	83	WUS	23
IND	84	BYP	24
NEL	85	NL	15
SSA	86	LC/RNL	06
ESA	87	IL/POC	17
HTS	88	SA	28
HTJ	89	SFE	29
VTS	8A	SM	2A
PLD	8B	CU2/CSP	2B
PLU	8C	MFA	2C
RI	8D	SPS	09
SS2	8E	SMM/RPT	0A
SS3	8F	CU1	1B
DCS	90		30
PU1	91		31
PU2	92	CC/UBS	1A
STS	93	IR	33
CCH	94	PN/PP	34
MW	95	RS/TRN	35
SPA	96	UC/NBS	36
EPA	97	GE	08

ASCII Character Data		EBCDIC Translation	
Character	Hex Value	Control/ Graphic	Hex Value
	98	SBS	38
	99	IT	39
	9A	RFF	3A
CSI	9B	CU3	3B
ST	9C	PF/SEL	04
OSC	9D	RES	14
PM	9E		3E
APC	9F		E1
NBSP	A0		41
¡	A1		42
¢	A2		43
£	A3		44
¤	A4		45
¥	A5		46
|	A6		47
§	A7		48
	A8		49
©	A9		51
ª	AA		52
«	AB		53
¬	AC		54
-(SHY)	AD		55
®	AE		56
_	AF		57
°	B0		58
±	B1		59
²	B2		62
³	B3		63
´	B4		64
µ	B5		65
¶	B6		66
·	B7		67
¸	B8		68

ASCII Character Data		EBCDIC Translation	
Character	Hex Value	Control/ Graphic	Hex Value
¹	B9		69
º	BA		70
»	BB		71
¼	BC		72
½	BD		73
¾	BE		74
¿	BF		75
À	C0		76
Á	C1		77
Â	C2		78
Ã	C3		80
Ä	C4		8A
Å	C5		8B
Æ	C6		8C
Ç	C7		8D
È	C8		8E
É	C9		8F
Ê	CA		90
Ë	CB	|	6A
Ì	CC		9B
Í	CD		9C
Î	CE		9D
Ï	CF		9E
Ð	D0		9F
Ñ	D1		A0
Ò	D2		AA
Ó	D3		AB
Ô	D4		AC
Õ	D5	¢	4A
Ö	D6		AE
	D7		AF
Ø	D8		B0
Ù	D9		B1

ASCII Character Data		EBCDIC Translation	
Character	Hex Value	Control/ Graphic	Hex Value
Ú	DA		B2
Û	DB		B3
Ü	DC		B4
Ý	DD		B5
Þ	DE		B6
ß	DF		B7
à	E0		B8
á	E1		B9
â	E2		BA
ã	E3		BB
ä	E4		BC
å	E5		A1
æ	E6		BE
ç	E7		BF
è	E8		CA
é	E9		CB
ê	EA		CC
ë	EB		CD
ì	EC		CE
í	ED		CF
î	EE		DA
ï	EF		DB
ð	F0		DC
ñ	F1		DD
ò	F2		DE
ó	F3		DF
ô	F4		EA
õ	F5		EB
ö	F6		EC
	F7		ED
ø	F8		EE
ù	F9		EF
ú	FA		FA

ASCII Character Data		EBCDIC Translation	
Character	**Hex Value**	**Control/ Graphic**	**Hex Value**
û	FB		FB
ü	FC		FC
ý	FD		FD
þ	FE		FE
ÿ	FF	EO	FF

CP37USA to ISO 8859-1 8-bit ASCII

The following table shows IBM Code Page 37(US EBCDIC) to ISO 8859-1 translation.

CP37USA Character Data		ISO 8859-1 Translation	
Character	Hex Value	Control/ Graphic	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
SEL	04	ST	9C
HT	05	HT	09
RNL	06	SSA	86
DEL	07	DEL	7F
GE	08	EPA	97
SPS	09	RI	8D
RPT	0A	SS2	8E
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO	0E	SO/LS1	0E
SI	0F	SI/LS0	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
RES/ENP	14	OSC	9D
NL	15	NEL	85
BS	16	BS	08
POC	17	ESA	87

CP37USA Character Data		ISO 8859-1 Translation	
Character	Hex Value	Control/ Graphic	Hex Value
CAN	18	CAN	18
EM	19	EM	19
UBS	1A	PU2	92
CU1	1B	SS3	8F
IFS	1C	IFS	1C
IGS	1D	GS	1D
IRS	1E	RS	1E
IUS	1F	US	1F
DS	20	DS	80
SOS	21		81
FS	22	BPH	82
WUS	23	NBH	83
BYP/INP	24	IND	84
LF	25	LF	0A
ETB	26	ETB	17
ESC	27	ESC	1B
SA	28	HTS	88
SFE	29	HTJ	89
SM/SW	2A	VTS	8A
CSP	2B	PLD	8B
MFA	2C	PLU	8C
ENQ	2D	ENQ	05
ACK	2E	ACK	06
BEL	2F	BEL	07

CP37USA Character Data		ISO 8859-1 Translation	
Character	Hex Value	Control/ Graphic	Hex Value
	30	DCS	90
	31	PU1	91
	32	SYN	16
IR	33	STS	93
PP	34	CCH	94
TRN	35	MW	95
NBS	36	SPA	96
EOT	37	EOT	04
SBS	38	SOS	98
IT	39		99
RFF	3A	SCI	9A
CU3	3B	CSI	9B
DC4	3C	DC4	14
NAK	3D	NAK	15
	3E	PM	9E
SUB	3F	SUB	1A
Space	40	Space	20
NBSP	41	NBSP	A0
â	42	â	E2
ä	43	ä	E4
à	44	à	E0
á	45	á	E1
ã	46	ã	E3
å	47	å	E5
ç	48	ç	E7
ñ	49	ñ	F1
¢	4A	¢	A2
.	4B	.	2E

CP37USA Character Data		ISO 8859-1 Translation	
Character	Hex Value	Control/ Graphic	Hex Value
<	4C	<	3C
(	4D	(	28
+	4E	+	2B
|	4F	|	7C
&	50	&	26
é	51	é	E9
ê	52	ê	EA
ë	53	ë	EB
è	54	è	E8
í	55	í	ED
î	56	î	EE
ï	57	ï	EF
ì	58	ì	EC
ß	59	ß	DF
!	5A	!	21
$	5B	$	24
*	5C	*	2A
)	5D	)	29
;	5E	;	3B
¬	5F	¬	AC
-	60	-	2D
/	61	/	2F
Â	62	Â	C2
Ä	63	Ä	C4
À	64	À	C0
Á	65	Á	C1
Ã	66	Ã	C3
Å	67	Å	C5

CP37USA Character Data		ISO 8859-1 Translation	
Character	Hex Value	Control/ Graphic	Hex Value
Ç	68	Ç	C7
Ñ	69	Ñ	D1
¦	6A	¦	A6
,	6B	,	2C
%	6C	%	25
_	6D	_	5F
>	6E	>	3E
?	6F	?	3F
ø	70	ø	F8
É	71	É	C9
Ê	72	Ê	CA
Ë	73	Ë	CB
È	74	È	C8
Í	75	Í	CD
Î	76	Î	CE
Ï	77	Ï	CF
Ì	78	Ì	CC
`	79	`	60
:	7A	:	3A
#	7B	#	23
@	7C	@	40
'	7D	'	27
=	7E	=	3D
"	7F	"	22
Ø	80	Ø	D8
a	81	a	61
b	82	b	62
c	83	c	63

CP37USA Character Data		ISO 8859-1 Translation	
Character	Hex Value	Control/ Graphic	Hex Value
d	84	d	64
e	85	e	65
f	86	f	66
g	87	g	67
h	88	h	68
i	89	i	69
«	8A	«	AB
»	8B	»	BB
ð	8C	ð	F0
ý	8D	ý	FD
þ	8E	þ	FE
±	8F	±	B1
°	90	°	B0
j	91	j	6A
k	92	k	6B
l	93	l	6C
m	94	m	6D
n	95	n	6E
o	96	o	6F
p	97	p	70
q	98	q	71
r	99	r	72
ª	9A	ª	AA
º	9B	º	BA
æ	9C	æ	E6
¸	9D	¸	B8
Æ	9E	Æ	C6
¤	9F	¤	A4

CP37USA Character Data		ISO 8859-1 Translation	
Character	Hex Value	Control/ Graphic	Hex Value
µ	A0	µ	B5
~	A1	~	7E
s	A2	s	73
t	A3	t	74
u	A4	u	75
v	A5	v	76
w	A6	w	77
x	A7	x	78
y	A8	y	79
z	A9	z	7A
¡	AA	¡	A1
¿	AB	¿	BF
Ð	AC	Ð	D0
Ý	AD	Ý	DD
Þ	AE	Þ	DE
®	AF	®	AE
^	B0	^	5E
£	B1	£	A3
¥	B2	¥	A5
·	B3	·	B7
©	B4	©	A9
§	B5	§	A7
¶	B6	¶	B6
¼	B7	¼	BC
½	B8	½	BD
¾	B9	¾	BE
[	BA	[	5B
]	BB	]	5D

CP37USA Character Data		ISO 8859-1 Translation	
Character	Hex Value	Control/ Graphic	Hex Value
_	BC	_	AF
¨	BD	¨	A8
´	BE	´	B4
×	BF	×	D7
{	C0	{	7B
A	C1	A	41
B	C2	B	42
C	C3	C	43
D	C4	D	44
E	C5	E	45
F	C6	F	46
G	C7	G	47
H	C8	H	48
I	C9	I	49
-	CA	-	AD
ô	CB	ô	F4
ö	CC	ö	F6
ò	CD	ò	F2
ó	CE	ó	F3
õ	CF	õ	F5
}	D0	}	7D
J	D1	J	4A
K	D2	K	4B
L	D3	L	4C
M	D4	M	4D
N	D5	N	4E
O	D6	O	4F
P	D7	P	50

CP37USA Character Data		ISO 8859-1 Translation	
Character	Hex Value	Control/ Graphic	Hex Value
Q	D8	Q	51
R	D9	R	52
¹	DA	¹	B9
û	DB	û	FB
ü	DC	ü	FC
ù	DD	ù	F9
ú	DE	ú	FA
ÿ	DF	ÿ	FF
\	E0	\	5C
÷	E1	÷	F7
S	E2	S	53
T	E3	T	54
U	E4	U	55
V	E5	V	56
W	E6	W	57
X	E7	X	58
Y	E8	Y	59
Z	E9	Z	5A
²	EA	²	B2
Ô	EB	Ô	D4
Ö	EC	Ö	D6
Ò	ED	Ò	D2
Ó	EE	Ó	D3
Õ	EF	Õ	D5
0	F0	0	30
1	F1	1	31
2	F2	2	32
3	F3	3	33

CP37USA Character Data		ISO 8859-1 Translation	
Character	Hex Value	Control/ Graphic	Hex Value
4	F4	4	34
5	F5	5	35
6	F6	6	36
7	F7	7	37
8	F8	8	38
9	F9	9	39
³	FA	³	B3
Û	FB	Û	DB
Ü	FC	Ü	DC
Ù	FD	Ù	D9
Ú	FE	Ú	DA
APC	FF	APC	9F

CP297FRANCE to ISO 8859-1 8-bit ASCII

The following table shows IBM Code Page 297(French EBCDIC) to ISO 8859-1 translation.

CP297FRANCE Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
SEL	04	ST	9C
HT	05	HT	09
RNL	06	SSA	86
DEL	07	DEL	7F
GE	08	EPA	97
SPS	09	RI	8D
RPT	0A	SS2	8E
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO	0E	SO/LS1	0E
SI	0F	SI/LS0	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
RES/ENP	14	OSC	9D
NL	15	NEL	85
BS	16	BS	08

CP297FRANCE Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
POC	17	ESA	87
CAN	18	CAN	18
EM	19	EM	19
UBS	1A	PU2	92
CU1	1B	SS3	8F
IFS	1C	IFS	1C
IGS	1D	GS	1D
IRS	1E	RS	1E
IUS	1F	US	1F
DS	20	DS	80
SOS	21		81
FS	22	BPH	82
WUS	23	NBH	83
BYP/INP	24	IND	84
LF	25	LF	0A
ETB	26	ETB	17
ESC	27	ESC	1B
SA	28	HTS	88
SFE	29	HTJ	89
SM/SW	2A	VTS	8A
CSP	2B	PLD	8B
MFA	2C	PLU	8C
ENQ	2D	ENQ	05

CP297FRANCE Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
ACK	2E	ACK	06
BEL	2F	BEL	07
	30	DCS	90
	31	PU1	91
	32	SYN	16
IR	33	STS	93
PP	34	CCH	94
TRN	35	MW	95
NBS	36	SPA	96
EOT	37	EOT	04
SBS	38	SOS	98
IT	39		99
RFF	3A	SCI	9A
CU3	3B	CSI	9B
DC4	3C	DC4	14
NAK	3D	NAK	15
	3E	PM	9E
SUB	3F	SUB	1A
Space	40	Space	20
NBSP	41	NBSP	A0
â	42	â	E2
ä	43	ä	E4
@	44	@	40
á	45	á	E1
ã	46	ã	E3
å	47	å	E5
\	48	\	5C
ñ	49	ñ	F1

CP297FRANCE Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
°	4A	°	B0
.	4B	.	2E
<	4C	<	3C
(	4D	(	28
+	4E	+	2B
!	4F	!	21
&	50	&	26
{	51	{	7B
ê	52	ê	EA
ë	53	ë	EB
}	54	}	7D
í	55	í	ED
î	56	î	EE
ï	57	ï	EF
ì	58	ì	EC
ß	59	ß	DF
§	5A	§	A7
$	5B	$	24
*	5C	*	2A
)	5D	)	29
;	5E	;	3B
^	5F	^	5E
-	60	-	2D
/	61	/	2F
Â	62	Â	C2
Ä	63	Ä	C4
À	64	À	C0
Á	65	Á	C1

CP297FRANCE Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
Ã	66	Ã	C3
Å	67	Å	C5
Ç	68	Ç	C7
Ñ	69	Ñ	D1
ù	6A	ù	F9
,	6B	,	2C
%	6C	%	25
_	6D	_	5F
>	6E	>	3E
?	6F	?	3F
ø	70	ø	F8
É	71	É	C9
Ê	72	Ê	CA
Ë	73	Ë	CB
È	74	È	C8
Í	75	Í	CD
Î	76	Î	CE
Ï	77	Ï	CF
Ì	78	Ì	CC
µ	79	µ	B5
:	7A	:	3A
£	7B	£	A3
à	7C	à	E0
'	7D	'	27
=	7E	=	3D
"	7F	"	22
Ø	80	Ø	D8
a	81	a	61

CP297FRANCE Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
b	82	b	62
c	83	c	63
d	84	d	64
e	85	e	65
f	86	f	66
g	87	g	67
h	88	h	68
i	89	i	69
«	8A	«	AB
»	8B	»	BB
ð	8C	ð	F0
ý	8D	ý	FD
þ	8E	þ	FE
±	8F	±	B1
[	90	[	5B
j	91	j	6A
k	92	k	6B
l	93	l	6C
m	94	m	6D
n	95	n	6E
o	96	o	6F
p	97	p	70
q	98	q	71
r	99	r	72
ª	9A	ª	AA
º	9B	º	BA
æ	9C	æ	E6
¸	9D	¸	B8

CP297FRANCE Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
Æ	9E	Æ	C6
¤	9F	¤	A4
`	A0	`	60
¨	A1	¨	A8
s	A2	s	73
t	A3	t	74
u	A4	u	75
v	A5	v	76
w	A6	w	77
x	A7	x	78
y	A8	y	79
z	A9	z	7A
¡	AA	¡	A1
¿	AB	¿	BF
Ð	AC	Ð	D0
Ý	AD	Ý	DD
Þ	AE	Þ	DE
®	AF	®	AE
¢	B0	¢	A2
#	B1	#	23
¥	B2	¥	A5
·	B3	·	B7
©	B4	©	A9
]	B5	]	5D
¶	B6	¶	B6
¼	B7	¼	BC
½	B8	½	BD
¾	B9	¾	BE

CP297FRANCE Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
¬	BA	¬	AC
|	BB	|	7C
_	BC	_	AF
~	BD	~	7E
´	BE	´	B4
×	BF	×	D7
é	C0	é	E9
A	C1	A	41
B	C2	B	42
C	C3	C	43
D	C4	D	44
E	C5	E	45
F	C6	F	46
G	C7	G	47
H	C8	H	48
I	C9	I	49
-	CA	-	AD
ô	CB	ô	F4
ö	CC	ö	F6
ò	CD	ò	F2
ó	CE	ó	F3
õ	CF	õ	F5
è	D0	è	E8
J	D1	J	4A
K	D2	K	4B
L	D3	L	4C
M	D4	M	4D
N	D5	N	4E

CP297FRANCE Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
O	D6	O	4F
P	D7	P	50
Q	D8	Q	51
R	D9	R	52
¹	DA	¹	B9
û	DB	û	FB
ü	DC	ü	FC
¦	DD	¦	A6
ú	DE	ú	FA
ÿ	DF	ÿ	FF
ç	E0	ç	E7
÷	E1	÷	F7
S	E2	S	53
T	E3	T	54
U	E4	U	55
V	E5	V	56
W	E6	W	57
X	E7	X	58
Y	E8	Y	59
Z	E9	Z	5A
²	EA	²	B2
Ô	EB	Ô	D4
Ö	EC	Ö	D6
Ò	ED	Ò	D2
Ó	EE	Ó	D3
Õ	EF	Õ	D5
0	F0	0	30
1	F1	1	31

CP297FRANCE Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
2	F2	2	32
3	F3	3	33
4	F4	4	34
5	F5	5	35
6	F6	6	36
7	F7	7	37
8	F8	8	38
9	F9	9	39
³	FA	³	B3
Û	FB	Û	DB
Ü	FC	Ü	DC
Ù	FD	Ù	D9
Ú	FE	Ú	DA
APC	FF	APC	9F

CP500GLOBAL to ISO 8859-1 8-bit ASCII

The following table shows IBM Code Page 500(Global EBCDIC) to ISO 8859-1 translation.

CP500GLOBAL Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
SEL	04	ST	9C
HT	05	HT	09
RNL	06	SSA	86
DEL	07	DEL	7F
GE	08	EPA	97
SPS	09	RI	8D
RPT	0A	SS2	8E
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO	0E	SO/LS1	0E
SI	0F	SI/LS0	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
RES/ENP	14	OSC	9D
NL	15	NEL	85
BS	16	BS	08
POC	17	ESA	87

CP500GLOBAL Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
UBS	1A	PU2	92
CU1	1B	SS3	8F
IFS	1C	IFS	1C
IGS	1D	GS	1D
IRS	1E	RS	1E
IUS	1F	US	1F
DS	20	DS	80
SOS	21		81
FS	22	BPH	82
WUS	23	NBH	83
BYP/INP	24	IND	84
LF	25	LF	0A
ETB	26	ETB	17
ESC	27	ESC	1B
SA	28	HTS	88
SFE	29	HTJ	89
SM/SW	2A	VTS	8A
CSP	2B	PLD	8B
MFA	2C	PLU	8C
ENQ	2D	ENQ	05
ACK	2E	ACK	06
BEL	2F	BEL	07

CP500GLOBAL Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
	30	DCS	90
	31	PU1	91
	32	SYN	16
IR	33	STS	93
PP	34	CCH	94
TRN	35	MW	95
NBS	36	SPA	96
EOT	37	EOT	04
SBS	38	SOS	98
IT	39		99
RFF	3A	SCI	9A
CU3	3B	CSI	9B
DC4	3C	DC4	14
NAK	3D	NAK	15
	3E	PM	9E
SUB	3F	SUB	1A
Space	40	Space	20
NBSP	41	NBSP	A0
â	42	â	E2
ä	43	ä	E4
à	44	à	E0
á	45	á	E1
ã	46	ã	E3
å	47	å	E5
ç	48	ç	E7
ñ	49	ñ	F1
[	4A	[	5B
.	4B	.	2E

CP500GLOBAL Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
<	4C	<	3C
(	4D	(	28
+	4E	+	2B
!	4F	!	21
&	50	&	26
é	51	é	E9
ê	52	ê	EA
ë	53	ë	EB
è	54	è	E8
í	55	í	ED
î	56	î	EE
ï	57	ï	EF
ì	58	ì	EC
ß	59	ß	DF
]	5A	]	5D
$	5B	$	24
*	5C	*	2A
)	5D	)	29
;	5E	;	3B
^	5F	^	5E
-	60	-	2D
/	61	/	2F
Â	62	Â	C2
Ä	63	Ä	C4
À	64	À	C0
Á	65	Á	C1
Ã	66	Ã	C3
Å	67	Å	C5

CP500GLOBAL Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
Ç	68	Ç	C7
Ñ	69	Ñ	D1
¦	6A	¦	A6
,	6B	,	2C
%	6C	%	25
_	6D	_	5F
>	6E	>	3E
?	6F	?	3F
ø	70	ø	F8
É	71	É	C9
Ê	72	Ê	CA
Ë	73	Ë	CB
È	74	È	C8
Í	75	Í	CD
Î	76	Î	CE
Ï	77	Ï	CF
Ì	78	Ì	CC
`	79	`	60
:	7A	:	3A
#	7B	#	23
@	7C	@	40
'	7D	'	27
=	7E	=	3D
"	7F	"	22
Ø	80	Ø	D8
a	81	a	61
b	82	b	62
c	83	c	63

CP500GLOBAL Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
d	84	d	64
e	85	e	65
f	86	f	66
g	87	g	67
h	88	h	68
i	89	i	69
«	8A	«	AB
»	8B	»	BB
ð	8C	ð	F0
ý	8D	ý	FD
þ	8E	þ	FE
±	8F	±	B1
°	90	°	B0
j	91	j	6A
k	92	k	6B
l	93	l	6C
m	94	m	6D
n	95	n	6E
o	96	o	6F
p	97	p	70
q	98	q	71
r	99	r	72
ª	9A	ª	AA
º	9B	º	BA
æ	9C	æ	E6
¸	9D	¸	B8
Æ	9E	Æ	C6
¤	9F	¤	A4

CP500GLOBAL Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
µ	A0	µ	B5
~	A1	~	7E
s	A2	s	73
t	A3	t	74
u	A4	u	75
v	A5	v	76
w	A6	w	77
x	A7	x	78
y	A8	y	79
z	A9	z	7A
¡	AA	¡	A1
¿	AB	¿	BF
Ð	AC	Ð	D0
Ý	AD	Ý	DD
Þ	AE	Þ	DE
®	AF	®	AE
¢	B0	¢	A2
£	B1	£	A3
¥	B2	¥	A5
·	B3	·	B7
©	B4	©	A9
§	B5	§	A7
¶	B6	¶	B6
¼	B7	¼	BC
½	B8	½	BD
¾	B9	¾	BE
¬	BA	¬	AC
|	BB	|	7C

CP500GLOBAL Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
_	BC	_	AF
¨	BD	¨	A8
´	BE	´	B4
×	BF	×	D7
{	C0	{	7B
A	C1	A	41
B	C2	B	42
C	C3	C	43
D	C4	D	44
E	C5	E	45
F	C6	F	46
G	C7	G	47
H	C8	H	48
I	C9	I	49
-	CA	-	AD
ô	CB	ô	F4
ö	CC	ö	F6
ò	CD	ò	F2
ó	CE	ó	F3
õ	CF	õ	F5
}	D0	}	7D
J	D1	J	4A
K	D2	K	4B
L	D3	L	4C
M	D4	M	4D
N	D5	N	4E
O	D6	O	4F
P	D7	P	50

CP500GLOBAL Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
Q	D8	Q	51
R	D9	R	52
¹	DA	¹	B9
û	DB	û	FB
ü	DC	ü	FC
ù	DD	ù	F9
ú	DE	ú	FA
ÿ	DF	ÿ	FF
\	E0	\	5C
÷	E1	÷	F7
S	E2	S	53
T	E3	T	54
U	E4	U	55
V	E5	V	56
W	E6	W	57
X	E7	X	58
Y	E8	Y	59
Z	E9	Z	5A
²	EA	²	B2
Ô	EB	Ô	D4
Ö	EC	Ö	D6
Ò	ED	Ò	D2
Ó	EE	Ó	D3
Õ	EF	Õ	D5
0	F0	0	30
1	F1	1	31
2	F2	2	32
3	F3	3	33

CP500GLOBAL Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
4	F4	4	34
5	F5	5	35
6	F6	6	36
7	F7	7	37
8	F8	8	38
9	F9	9	39
³	FA	³	B3
Û	FB	Û	DB
Ü	FC	Ü	DC
Ù	FD	Ù	D9
Ú	FE	Ú	DA
APC	FF	APC	9F

CP273GERMANY to ISO 8859-1 8-bit ASCII

The following table shows IBM Code Page 273(GERMANY EBCDIC) to ISO 8859-1 translation.

CP273GERMANY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
SEL	04	ST	9C
HT	05	HT	09
RNL	06	SSA	86
DEL	07	DEL	7F
GE	08	EPA	97
SPS	09	RI	8D
RPT	0A	SS2	8E
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO	0E	SO/LS1	0E
SI	0F	SI/LS0	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
RES/ENP	14	OSC	9D
NL	15	NEL	85
BS	16	BS	08
POC	17	ESA	87

CP273GERMANY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
UBS	1A	PU2	92
CU1	1B	SS3	8F
IFS	1C	IFS	1C
IGS	1D	GS	1D
IRS	1E	RS	1E
IUS	1F	US	1F
DS	20	DS	80
SOS	21		81
FS	22	BPH	82
WUS	23	NBH	83
BYP/INP	24	IND	84
LF	25	LF	0A
ETB	26	ETB	17
ESC	27	ESC	1B
SA	28	HTS	88
SFE	29	HTJ	89
SM/SW	2A	VTS	8A
CSP	2B	PLD	8B
MFA	2C	PLU	8C
ENQ	2D	ENQ	05
ACK	2E	ACK	06
BEL	2F	BEL	07

CP273GERMANY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
	30	DCS	90
	31	PU1	91
	32	SYN	16
IR	33	STS	93
PP	34	CCH	94
TRN	35	MW	95
NBS	36	SPA	96
EOT	37	EOT	04
SBS	38	SOS	98
IT	39		99
RFF	3A	SCI	9A
CU3	3B	CSI	9B
DC4	3C	DC4	14
NAK	3D	NAK	15
	3E	PM	9E
SUB	3F	SUB	1A
Space	40	Space	20
NBSP	41	NBSP	A0
â	42	â	E2
{	43	{	7B
à	44	à	E0
á	45	á	E1
ã	46	ã	E3
å	47	å	E5
ç	48	ç	E7
ñ	49	ñ	F1
Ä	4A	Ä	C4
.	4B	.	2E

CP273GERMANY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
<	4C	<	3C
(	4D	(	28
+	4E	+	2B
!	4F	!	21
&	50	&	26
é	51	é	E9
ê	52	ê	EA
ë	53	ë	EB
è	54	è	E8
í	55	í	ED
î	56	î	EE
ï	57	ï	EF
ì	58	ì	EC
~	59	~	7E
Ü	5A	Ü	DC
$	5B	$	24
*	5C	*	2A
)	5D	)	29
;	5E	;	3B
^	5F	^	5E
-	60	-	2D
/	61	/	2F
Â	62	Â	C2
[	63	[	5B
À	64	À	C0
Á	65	Á	C1
Ã	66	Ã	C3
Å	67	Å	C5

CP273GERMANY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
Ç	68	Ç	C7
Ñ	69	Ñ	D1
ö	6A	ö	F6
,	6B	,	2C
%	6C	%	25
_	6D	_	5F
>	6E	>	3E
?	6F	?	3F
ø	70	ø	F8
É	71	É	C9
Ê	72	Ê	CA
Ë	73	Ë	CB
È	74	È	C8
Í	75	Í	CD
Î	76	Î	CE
Ï	77	Ï	CF
Ì	78	Ì	CC
`	79	`	60
:	7A	:	3A
#	7B	#	23
§	7C	§	A7
'	7D	'	27
=	7E	=	3D
"	7F	"	22
Ø	80	Ø	D8
a	81	a	61
b	82	b	62
c	83	c	63

CP273GERMANY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
d	84	d	64
e	85	e	65
f	86	f	66
g	87	g	67
h	88	h	68
i	89	i	69
«	8A	«	AB
»	8B	»	BB
ð	8C	ð	F0
ý	8D	ý	FD
þ	8E	þ	FE
±	8F	±	B1
°	90	°	B0
j	91	j	6A
k	92	k	6B
l	93	l	6C
m	94	m	6D
n	95	n	6E
o	96	o	6F
p	97	p	70
q	98	q	71
r	99	r	72
ª	9A	ª	AA
º	9B	º	BA
æ	9C	æ	E6
¸	9D	¸	B8
Æ	9E	Æ	C6
¤	9F	¤	A4

CP273GERMANY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
µ	A0	µ	B5
ß	A1	ß	DF
s	A2	s	73
t	A3	t	74
u	A4	u	75
v	A5	v	76
w	A6	w	77
x	A7	x	78
y	A8	y	79
z	A9	z	7A
¡	AA	¡	A1
¿	AB	¿	BF
Ð	AC	Ð	D0
Ý	AD	Ý	DD
Þ	AE	Þ	DE
®	AF	®	AE
¢	B0	¢	A2
£	B1	£	A3
¥	B2	¥	A5
·	B3	·	B7
©	B4	©	A9
@	B5	@	40
¶	B6	¶	B6
¼	B7	¼	BC
½	B8	½	BD
¾	B9	¾	BE
¬	BA	¬	AC
|	BB	|	7C

CP273GERMANY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
_	BC	_	AF
¨	BD	¨	A8
´	BE	´	B4
×	BF	×	D7
ä	C0	ä	E4
A	C1	A	41
B	C2	B	42
C	C3	C	43
D	C4	D	44
E	C5	E	45
F	C6	F	46
G	C7	G	47
H	C8	H	48
I	C9	I	49
-	CA	-	AD
ô	CB	ô	F4
¦	CC	¦	A6
ò	CD	ò	F2
ó	CE	ó	F3
õ	CF	õ	F5
ü	D0	ü	FC
J	D1	J	4A
K	D2	K	4B
L	D3	L	4C
M	D4	M	4D
N	D5	N	4E
O	D6	O	4F
P	D7	P	50

CP273GERMANY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
Q	D8	Q	51
R	D9	R	52
¹	DA	¹	B9
û	DB	û	FB
}	DC	}	7D
ù	DD	ù	F9
ú	DE	ú	FA
ÿ	DF	ÿ	FF
Ö	E0	Ö	D6
÷	E1	÷	F7
S	E2	S	53
T	E3	T	54
U	E4	U	55
V	E5	V	56
W	E6	W	57
X	E7	X	58
Y	E8	Y	59
Z	E9	Z	5A
²	EA	²	B2
Ô	EB	Ô	D4
Ö	EC	\	5C
Ò	ED	Ò	D2
Ó	EE	Ó	D3
Õ	EF	Õ	D5
0	F0	0	30
1	F1	1	31
2	F2	2	32
3	F3	3	33

CP273GERMANY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
4	F4	4	34
5	F5	5	35
6	F6	6	36
7	F7	7	37
8	F8	8	38
9	F9	9	39
³	FA	³	B3
Û	FB	Û	DB
]	FC	]	5D
Ù	FD	Ù	D9
Ú	FE	Ú	DA
APC	FF	APC	9F

CP277DENMARK to ISO 8859-1 8-bit ASCII

The following table shows IBM Code Page 277(Denmark EBCDIC) to ISO 8859-1 translation.

CP277DENMARK Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
SEL	04	ST	9C
HT	05	HT	09
RNL	06	SSA	86
DEL	07	DEL	7F
GE	08	EPA	97
SPS	09	RI	8D
RPT	0A	SS2	8E
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO	0E	SO/LS1	0E
SI	0F	SI/LS0	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
RES/ENP	14	OSC	9D
NL	15	NEL	85
BS	16	BS	08
POC	17	ESA	87

CP277DENMARK Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
UBS	1A	PU2	92
CU1	1B	SS3	8F
IFS	1C	IFS	1C
IGS	1D	GS	1D
IRS	1E	RS	1E
IUS	1F	US	1F
DS	20	DS	80
SOS	21		81
FS	22	BPH	82
WUS	23	NBH	83
BYP/INP	24	IND	84
LF	25	LF	0A
ETB	26	ETB	17
ESC	27	ESC	1B
SA	28	HTS	88
SFE	29	HTJ	89
SM/SW	2A	VTS	8A
CSP	2B	PLD	8B
MFA	2C	PLU	8C
ENQ	2D	ENQ	05
ACK	2E	ACK	06
BEL	2F	BEL	07

CP277DENMARK Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
	30	DCS	90
	31	PU1	91
	32	SYN	16
IR	33	STS	93
PP	34	CCH	94
TRN	35	MW	95
NBS	36	SPA	96
EOT	37	EOT	04
SBS	38	SOS	98
IT	39		99
RFF	3A	SCI	9A
CU3	3B	CSI	9B
DC4	3C	DC4	14
NAK	3D	NAK	15
	3E	PM	9E
SUB	3F	SUB	1A
Space	40	Space	20
NBSP	41	NBSP	A0
â	42	â	E2
ä	43	ä	E4
à	44	à	E0
á	45	á	E1
ã	46	ã	E3
}	47	}	7D
ç	48	ç	E7
ñ	49	ñ	F1
#	4A	#	23
.	4B	.	2E

CP277DENMARK Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
<	4C	<	3C
(	4D	(	28
+	4E	+	2B
!	4F	!	21
&	50	&	26
é	51	é	E9
ê	52	ê	EA
ë	53	ë	EB
è	54	è	E8
í	55	í	ED
î	56	î	EE
ï	57	ï	EF
ì	58	ì	EC
ß	59	ß	DF
¤	5A	¤	A4
Å	5B	Å	C5
*	5C	*	2A
)	5D	)	29
;	5E	;	3B
^	5F	^	5E
-	60	-	2D
/	61	/	2F
Â	62	Â	C2
Ä	63	Ä	C4
À	64	À	C0
Á	65	Á	C1
Ã	66	Ã	C3
$	67	$	24

CP277DENMARK Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
Ç	68	Ç	C7
Ñ	69	Ñ	D1
ø	6A	ø	F8
,	6B	,	2C
%	6C	%	25
_	6D	_	5F
>	6E	>	3E
?	6F	?	3F
¦	70	¦	A6
É	71	É	C9
Ê	72	Ê	CA
Ë	73	Ë	CB
È	74	È	C8
Í	75	Í	CD
Î	76	Î	CE
Ï	77	Ï	CF
Ì	78	Ì	CC
`	79	`	60
:	7A	:	3A
Æ	7B	Æ	C6
Ø	7C	Ø	D8
'	7D	'	27
=	7E	=	3D
"	7F	"	22
@	80	@	40
a	81	a	61
b	82	b	62
c	83	c	63

CP277DENMARK Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
d	84	d	64
e	85	e	65
f	86	f	66
g	87	g	67
h	88	h	68
i	89	i	69
«	8A	«	AB
»	8B	»	BB
ð	8C	ð	F0
ý	8D	ý	FD
þ	8E	þ	FE
±	8F	±	B1
°	90	°	B0
j	91	j	6A
k	92	k	6B
l	93	l	6C
m	94	m	6D
n	95	n	6E
o	96	o	6F
p	97	p	70
q	98	q	71
r	99	r	72
ª	9A	ª	AA
º	9B	º	BA
{	9C	{	7B
¸	9D	¸	B8
[	9E	[	5B
]	9F	]	5D

CP277DENMARK Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
µ	A0	µ	B5
ü	A1	ü	FC
s	A2	s	73
t	A3	t	74
u	A4	u	75
v	A5	v	76
w	A6	w	77
x	A7	x	78
y	A8	y	79
z	A9	z	7A
¡	AA	¡	A1
¿	AB	¿	BF
Ð	AC	Ð	D0
Ý	AD	Ý	DD
Þ	AE	Þ	DE
®	AF	®	AE
¢	B0	¢	A2
£	B1	£	A3
¥	B2	¥	A5
·	B3	·	B7
©	B4	©	A9
§	B5	§	A7
¶	B6	¶	B6
¼	B7	¼	BC
½	B8	½	BD
¾	B9	¾	BE
¬	BA	¬	AC
|	BB	|	7C

CP277DENMARK Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
_	BC	_	AF
¨	BD	¨	A8
´	BE	´	B4
×	BF	×	D7
æ	C0	æ	E6
A	C1	A	41
B	C2	B	42
C	C3	C	43
D	C4	D	44
E	C5	E	45
F	C6	F	46
G	C7	G	47
H	C8	H	48
I	C9	I	49
-	CA	-	AD
ô	CB	ô	F4
ö	CC	ö	F6
ò	CD	ò	F2
ó	CE	ó	F3
õ	CF	õ	F5
å	D0	å	E5
J	D1	J	4A
K	D2	K	4B
L	D3	L	4C
M	D4	M	4D
N	D5	N	4E
O	D6	O	4F
P	D7	P	50

CP277DENMARK Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
Q	D8	Q	51
R	D9	R	52
¹	DA	¹	B9
û	DB	û	FB
~	DC	~	7E
ù	DD	ù	F9
ú	DE	ú	FA
ÿ	DF	ÿ	FF
\	E0	\	5C
÷	E1	÷	F7
S	E2	S	53
T	E3	T	54
U	E4	U	55
V	E5	V	56
W	E6	W	57
X	E7	X	58
Y	E8	Y	59
Z	E9	Z	5A
²	EA	²	B2
Ô	EB	Ô	D4
Ö	EC	Ö	D6
Ò	ED	Ò	D2
Ó	EE	Ó	D3
Õ	EF	Õ	D5
0	F0	0	30
1	F1	1	31
2	F2	2	32
3	F3	3	33

CP277DENMARK Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
4	F4	4	34
5	F5	5	35
6	F6	6	36
7	F7	7	37
8	F8	8	38
9	F9	9	39
³	FA	³	B3
Û	FB	Û	DB
Ü	FC	Ü	DC
Ù	FD	Ù	D9
Ú	FE	Ú	DA
APC	FF	APC	9F

CP278SWEDEN to ISO 8859-1 8-bit ASCII

The following table shows IBM Code Page 278(Sweden EBCDIC) to ISO 8859-1 translation.

CP278SWEDEN Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
SEL	04	ST	9C
HT	05	HT	09
RNL	06	SSA	86
DEL	07	DEL	7F
GE	08	EPA	97
SPS	09	RI	8D
RPT	0A	SS2	8E
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO	0E	SO/LS1	0E
SI	0F	SI/LS0	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
RES/ENP	14	OSC	9D
NL	15	NEL	85
BS	16	BS	08
POC	17	ESA	87

CP278SWEDEN Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
UBS	1A	PU2	92
CU1	1B	SS3	8F
IFS	1C	IFS	1C
IGS	1D	GS	1D
IRS	1E	RS	1E
IUS	1F	US	1F
DS	20	DS	80
SOS	21		81
FS	22	BPH	82
WUS	23	NBH	83
BYP/INP	24	IND	84
LF	25	LF	0A
ETB	26	ETB	17
ESC	27	ESC	1B
SA	28	HTS	88
SFE	29	HTJ	89
SM/SW	2A	VTS	8A
CSP	2B	PLD	8B
MFA	2C	PLU	8C
ENQ	2D	ENQ	05
ACK	2E	ACK	06
BEL	2F	BEL	07

CP278SWEDEN Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
	30	DCS	90
	31	PU1	91
	32	SYN	16
IR	33	STS	93
PP	34	CCH	94
TRN	35	MW	95
NBS	36	SPA	96
EOT	37	EOT	04
SBS	38	SOS	98
IT	39		99
RFF	3A	SCI	9A
CU3	3B	CSI	9B
DC4	3C	DC4	14
NAK	3D	NAK	15
	3E	PM	9E
SUB	3F	SUB	1A
Space	40	Space	20
NBSP	41	NBSP	A0
â	42	â	E2
{	43	{	7B
à	44	à	E0
á	45	á	E1
ã	46	ã	E3
}	47	}	7D
ç	48	ç	E7
ñ	49	ñ	F1
§	4A	§	A7
.	4B	.	2E

CP278SWEDEN Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
<	4C	<	3C
(	4D	(	28
+	4E	+	2B
!	4F	!	21
&	50	&	26
`	51	`	60
ê	52	ê	EA
ë	53	ë	EB
è	54	è	E8
í	55	í	ED
î	56	î	EE
ï	57	ï	EF
ì	58	ì	EC
ß	59	ß	DF
¤	5A	¤	A4
Å	5B	Å	C5
*	5C	*	2A
)	5D	)	29
;	5E	;	3B
^	5F	^	5E
-	60	-	2D
/	61	/	2F
Â	62	Â	C2
#	63	#	23
À	64	À	C0
Á	65	Á	C1
Ã	66	Ã	C3
$	67	$	24

CP278SWEDEN Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
Ç	68	Ç	C7
Ñ	69	Ñ	D1
ö	6A	ö	F6
,	6B	,	2C
%	6C	%	25
_	6D	_	5F
>	6E	>	3E
?	6F	?	3F
ø	70	ø	F8
\	71	\	5C
Ê	72	Ê	CA
Ë	73	Ë	CB
È	74	È	C8
Í	75	Í	CD
Î	76	Î	CE
Ï	77	Ï	CF
Ì	78	Ì	CC
é	79	é	E9
:	7A	:	3A
Ä	7B	Ä	C4
Ö	7C	Ö	D6
'	7D	'	27
=	7E	=	3D
"	7F	"	22
Ø	80	Ø	D8
a	81	a	61
b	82	b	62
c	83	c	63

CP278SWEDEN Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
d	84	d	64
e	85	e	65
f	86	f	66
g	87	g	67
h	88	h	68
i	89	i	69
«	8A	«	AB
»	8B	»	BB
ð	8C	ð	F0
ý	8D	ý	FD
þ	8E	þ	FE
±	8F	±	B1
°	90	°	B0
j	91	j	6A
k	92	k	6B
l	93	l	6C
m	94	m	6D
n	95	n	6E
o	96	o	6F
p	97	p	70
q	98	q	71
r	99	r	72
ª	9A	ª	AA
º	9B	º	BA
æ	9C	æ	E6
¸	9D	¸	B8
Æ	9E	Æ	C6
]	9F	]	5D

CP278SWEDEN Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
µ	A0	µ	B5
ü	A1	ü	FC
s	A2	s	73
t	A3	t	74
u	A4	u	75
v	A5	v	76
w	A6	w	77
x	A7	x	78
y	A8	y	79
z	A9	z	7A
¡	AA	¡	A1
¿	AB	¿	BF
Ð	AC	Ð	D0
Ý	AD	Ý	DD
Þ	AE	Þ	DE
®	AF	®	AE
¢	B0	¢	A2
£	B1	£	A3
¥	B2	¥	A5
·	B3	·	B7
©	B4	©	A9
[	B5	[	5B
¶	B6	¶	B6
¼	B7	¼	BC
½	B8	½	BD
¾	B9	¾	BE
¬	BA	¬	AC
|	BB	|	7C

CP278SWEDEN Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
_	BC	_	AF
¨	BD	¨	A8
´	BE	´	B4
×	BF	×	D7
ä	C0	ä	E4
A	C1	A	41
B	C2	B	42
C	C3	C	43
D	C4	D	44
E	C5	E	45
F	C6	F	46
G	C7	G	47
H	C8	H	48
I	C9	I	49
-	CA	-	AD
ô	CB	ô	F4
¦	CC	¦	A6
ò	CD	ò	F2
ó	CE	ó	F3
õ	CF	õ	F5
å	D0	å	E5
J	D1	J	4A
K	D2	K	4B
L	D3	L	4C
M	D4	M	4D
N	D5	N	4E
O	D6	O	4F
P	D7	P	50

CP278SWEDEN Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
Q	D8	Q	51
R	D9	R	52
¹	DA	¹	B9
û	DB	û	FB
~	DC	~	7E
ù	DD	ù	F9
ú	DE	ú	FA
ÿ	DF	ÿ	FF
É	E0	É	C9
÷	E1	÷	F7
S	E2	S	53
T	E3	T	54
U	E4	U	55
V	E5	V	56
W	E6	W	57
X	E7	X	58
Y	E8	Y	59
Z	E9	Z	5A
²	EA	²	B2
Ô	EB	Ô	D4
@	EC	@	40
Ò	ED	Ò	D2
Ó	EE	Ó	D3
Õ	EF	Õ	D5
0	F0	0	30
1	F1	1	31
2	F2	2	32
3	F3	3	33

CP278SWEDEN Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
4	F4	4	34
5	F5	5	35
6	F6	6	36
7	F7	7	37
8	F8	8	38
9	F9	9	39
³	FA	³	B3
Û	FB	Û	DB
Ü	FC	Ü	DC
Ù	FD	Ù	D9
Ú	FE	Ú	DA
APC	FF	APC	9F

CP280ITALY to ISO 8859-1 8-bit ASCII

The following table shows IBM Code Page 280(Italy EBCDIC) to ISO 8859-1 translation.

CP280ITALY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
SEL	04	ST	9C
HT	05	HT	09
RNL	06	SSA	86
DEL	07	DEL	7F
GE	08	EPA	97
SPS	09	RI	8D
RPT	0A	SS2	8E
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO	0E	SO/LS1	0E
SI	0F	SI/LS0	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
RES/ENP	14	OSC	9D
NL	15	NEL	85
BS	16	BS	08
POC	17	ESA	87

CP280ITALY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
UBS	1A	PU2	92
CU1	1B	SS3	8F
IFS	1C	IFS	1C
IGS	1D	GS	1D
IRS	1E	RS	1E
IUS	1F	US	1F
DS	20	DS	80
SOS	21		81
FS	22	BPH	82
WUS	23	NBH	83
BYP/INP	24	IND	84
LF	25	LF	0A
ETB	26	ETB	17
ESC	27	ESC	1B
SA	28	HTS	88
SFE	29	HTJ	89
SM/SW	2A	VTS	8A
CSP	2B	PLD	8B
MFA	2C	PLU	8C
ENQ	2D	ENQ	05
ACK	2E	ACK	06
BEL	2F	BEL	07

CP280ITALY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
	30	DCS	90
	31	PU1	91
	32	SYN	16
IR	33	STS	93
PP	34	CCH	94
TRN	35	MW	95
NBS	36	SPA	96
EOT	37	EOT	04
SBS	38	SOS	98
IT	39		99
RFF	3A	SCI	9A
CU3	3B	CSI	9B
DC4	3C	DC4	14
NAK	3D	NAK	15
	3E	PM	9E
SUB	3F	SUB	1A
Space	40	Space	20
NBSP	41	NBSP	A0
â	42	â	E2
ä	43	ä	E4
{	44	{	7B
á	45	á	E1
ã	46	ã	E3
å	47	å	E5
\	48	\	5C
ñ	49	ñ	F1
°	4A	°	B0
.	4B	.	2E

CP280ITALY Data		ISO 8859-1 Translation	
Control/ Graphic	Hex Value	Character	Hex Value
<	4C	<	3C
(	4D	(	28
+	4E	+	2B
!	4F	!	21
&	50	&	26
]	51	]	5D
ê	52	ê	EA
ë	53	ë	EB
}	54	}	7D
í	55	í	ED
î	56	î	EE
ï	57	ï	EF
~	58	~	7E
ß	59	ß	DF
é	5A	é	E9
$	5B	$	24
*	5C	*	2A
)	5D	)	29
;	5E	;	3B
^	5F	^	5E
-	60	-	2D
/	61	/	2F
Â	62	Â	C2
Ä	63	Ä	C4
À	64	À	C0
Á	65	Á	C1
Ã	66	Ã	C3
Å	67	Å	C5

CP280ITALY Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ç	68	Ç	C7
Ñ	69	Ñ	D1
ò	6A	ò	F2
,	6B	,	2C
%	6C	%	25
_	6D	_	5F
>	6E	>	3E
?	6F	?	3F
ø	70	ø	F8
É	71	É	C9
Ê	72	Ê	CA
Ë	73	Ë	CB
È	74	È	C8
Í	75	Í	CD
Î	76	Î	CE
Ï	77	Ï	CF
Ì	78	Ì	CC
ù	79	ù	F9
:	7A	:	3A
£	7B	£	A3
§	7C	§	A7
'	7D	'	27
=	7E	=	3D
"	7F	"	22
Ø	80	Ø	D8
a	81	a	61
b	82	b	62
c	83	c	63

CP280ITALY Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
d	84	d	64
e	85	e	65
f	86	f	66
g	87	g	67
h	88	h	68
i	89	i	69
«	8A	«	AB
»	8B	»	BB
ð	8C	ð	F0
ý	8D	ý	FD
þ	8E	þ	FE
±	8F	±	B1
[	90	[	5B
j	91	j	6A
k	92	k	6B
l	93	l	6C
m	94	m	6D
n	95	n	6E
o	96	o	6F
p	97	p	70
q	98	q	71
r	99	r	72
ª	9A	ª	AA
º	9B	º	BA
æ	9C	æ	E6
¸	9D	¸	B8
Æ	9E	Æ	C6
¤	9F	¤	A4

CP280ITALY Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
µ	A0	µ	B5
ì	A1	ì	EC
s	A2	s	73
t	A3	t	74
u	A4	u	75
v	A5	v	76
w	A6	w	77
x	A7	x	78
y	A8	y	79
z	A9	z	7A
¡	AA	¡	A1
¿	AB	¿	BF
Ð	AC	Ð	D0
Ý	AD	Ý	DD
Þ	AE	Þ	DE
®	AF	®	AE
¢	B0	¢	A2
#	B1	#	23
¥	B2	¥	A5
·	B3	·	B7
©	B4	©	A9
@	B5	@	40
¶	B6	¶	B6
¼	B7	¼	BC
½	B8	½	BD
¾	B9	¾	BE
¬	BA	¬	AC
|	BB	|	7C

CP280ITALY Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
_	BC	_	AF
¨	BD	¨	A8
´	BE	´	B4
×	BF	×	D7
à	C0	à	E0
A	C1	A	41
B	C2	B	42
C	C3	C	43
D	C4	D	44
E	C5	E	45
F	C6	F	46
G	C7	G	47
H	C8	H	48
I	C9	I	49
-	CA	-	AD
ô	CB	ô	F4
ö	CC	ö	F6
¦	CD	¦	A6
ó	CE	ó	F3
õ	CF	õ	F5
è	D0	è	E8
J	D1	J	4A
K	D2	K	4B
L	D3	L	4C
M	D4	M	4D
N	D5	N	4E
O	D6	O	4F
P	D7	P	50

CP280ITALY Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
Q	D8	Q	51
R	D9	R	52
¹	DA	¹	B9
û	DB	û	FB
ü	DC	ü	FC
`	DD	`	60
ú	DE	ú	FA
ÿ	DF	ÿ	FF
ç	E0	ç	E7
÷	E1	÷	F7
S	E2	S	53
T	E3	T	54
U	E4	U	55
V	E5	V	56
W	E6	W	57
X	E7	X	58
Y	E8	Y	59
Z	E9	Z	5A
²	EA	²	B2
Ô	EB	Ô	D4
Ö	EC	Ö	D6
Ò	ED	Ò	D2
Ó	EE	Ó	D3
Õ	EF	Õ	D5
0	F0	0	30
1	F1	1	31
2	F2	2	32
3	F3	3	33

CP280ITALY Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
4	F4	4	34
5	F5	5	35
6	F6	6	36
7	F7	7	37
8	F8	8	38
9	F9	9	39
³	FA	³	B3
Û	FB	Û	DB
Ü	FC	Ü	DC
Ù	FD	Ù	D9
Ú	FE	Ú	DA
APC	FF	APC	9F

CP284SPAIN to ISO 8859-1 8-bit ASCII

The following table shows IBM Code Page 284(Spain EBCDIC) to ISO 8859-1 translation.

CP284SPAIN Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
SEL	04	ST	9C
HT	05	HT	09
RNL	06	SSA	86
DEL	07	DEL	7F
GE	08	EPA	97
SPS	09	RI	8D
RPT	0A	SS2	8E
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO	0E	SO/LS1	0E
SI	0F	SI/LS0	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
RES/ENP	14	OSC	9D
NL	15	NEL	85
BS	16	BS	08
POC	17	ESA	87

CP284SPAIN Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
UBS	1A	PU2	92
CU1	1B	SS3	8F
IFS	1C	IFS	1C
IGS	1D	GS	1D
IRS	1E	RS	1E
IUS	1F	US	1F
DS	20	DS	80
SOS	21		81
FS	22	BPH	82
WUS	23	NBH	83
BYP/INP	24	IND	84
LF	25	LF	0A
ETB	26	ETB	17
ESC	27	ESC	1B
SA	28	HTS	88
SFE	29	HTJ	89
SM/SW	2A	VTS	8A
CSP	2B	PLD	8B
MFA	2C	PLU	8C
ENQ	2D	ENQ	05
ACK	2E	ACK	06
BEL	2F	BEL	07

CP284SPAIN Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
	30	DCS	90
	31	PU1	91
	32	SYN	16
IR	33	STS	93
PP	34	CCH	94
TRN	35	MW	95
NBS	36	SPA	96
EOT	37	EOT	04
SBS	38	SOS	98
IT	39		99
RFF	3A	SCI	9A
CU3	3B	CSI	9B
DC4	3C	DC4	14
NAK	3D	NAK	15
	3E	PM	9E
SUB	3F	SUB	1A
Space	40	Space	20
NBSP	41	NBSP	A0
â	42	â	E2
ä	43	ä	E4
à	44	à	E0
á	45	á	E1
ã	46	ã	E3
å	47	å	E5
ç	48	ç	E7
¦	49	¦	A6
[	4A	[	5B
.	4B	.	2E
<	4C	<	3C
(	4D	(	28
+	4E	+	2B
|	4F	|	7C
&	50	&	26
é	51	é	E9
ê	52	ê	EA
ë	53	ë	EB
è	54	è	E8
í	55	í	ED
î	56	î	EE
ï	57	ï	EF
ì	58	ì	EC
ß	59	ß	DF
]	5A	]	5D
$	5B	$	24
*	5C	*	2A
)	5D	)	29
;	5E	;	3B
¬	5F	¬	AC
-	60	-	2D
/	61	/	2F
Â	62	Â	C2
Ä	63	Ä	C4
À	64	À	C0
Á	65	Á	C1
Ã	66	Ã	C3
Å	67	Å	C5

CP284SPAIN Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ç	68	Ç	C7
#	69	#	23
ñ	6A	ñ	F1
,	6B	,	2C
%	6C	%	25
_	6D	_	5F
>	6E	>	3E
?	6F	?	3F
ø	70	ø	F8
É	71	É	C9
Ê	72	Ê	CA
Ë	73	Ë	CB
È	74	È	C8
Í	75	Í	CD
Î	76	Î	CE
Ï	77	Ï	CF
Ì	78	Ì	CC
`	79	`	60
:	7A	:	3A
Ñ	7B	Ñ	D1
@	7C	@	40
'	7D	'	27
=	7E	=	3D
"	7F	"	22
Ø	80	Ø	D8
a	81	a	61
b	82	b	62
c	83	c	63

CP284SPAIN Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
d	84	d	64
e	85	e	65
f	86	f	66
g	87	g	67
h	88	h	68
i	89	i	69
«	8A	«	AB
»	8B	»	BB
ð	8C	ð	F0
ý	8D	ý	FD
þ	8E	þ	FE
±	8F	±	B1
°	90	°	B0
j	91	j	6A
k	92	k	6B
l	93	l	6C
m	94	m	6D
n	95	n	6E
o	96	o	6F
p	97	p	70
q	98	q	71
r	99	r	72
ª	9A	ª	AA
º	9B	º	BA
æ	9C	æ	E6
¸	9D	¸	B8
Æ	9E	Æ	C6
¤	9F	¤	A4

CP284SPAIN Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
µ	A0	µ	B5
¨	A1	¨	A8
s	A2	s	73
t	A3	t	74
u	A4	u	75
v	A5	v	76
w	A6	w	77
x	A7	x	78
y	A8	y	79
z	A9	z	7A
¡	AA	¡	A1
¿	AB	¿	BF
Ð	AC	Ð	D0
Ý	AD	Ý	DD
Þ	AE	Þ	DE
®	AF	®	AE
¢	B0	¢	A2
£	B1	£	A3
¥	B2	¥	A5
·	B3	·	B7
©	B4	©	A9
§	B5	§	A7
¶	B6	¶	B6
¼	B7	¼	BC
½	B8	½	BD
¾	B9	¾	BE
^	BA	^	5E
!	BB	!	21

CP284SPAIN Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
_	BC	_	AF
~	BD	~	7E
´	BE	´	B4
×	BF	×	D7
{	C0	{	7B
A	C1	A	41
B	C2	B	42
C	C3	C	43
D	C4	D	44
E	C5	E	45
F	C6	F	46
G	C7	G	47
H	C8	H	48
I	C9	I	49
-	CA	-	AD
ô	CB	ô	F4
ö	CC	ö	F6
ò	CD	ò	F2
ó	CE	ó	F3
õ	CF	õ	F5
}	D0	}	7D
J	D1	J	4A
K	D2	K	4B
L	D3	L	4C
M	D4	M	4D
N	D5	N	4E
O	D6	O	4F
P	D7	P	50

CP284SPAIN Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
Q	D8	Q	51
R	D9	R	52
¹	DA	¹	B9
û	DB	û	FB
ü	DC	ü	FC
ù	DD	ù	F9
ú	DE	ú	FA
ÿ	DF	ÿ	FF
\	E0	\	5C
÷	E1	÷	F7
S	E2	S	53
T	E3	T	54
U	E4	U	55
V	E5	V	56
W	E6	W	57
X	E7	X	58
Y	E8	Y	59
Z	E9	Z	5A
²	EA	²	B2
Ô	EB	Ô	D4
Ö	EC	Ö	D6
Ò	ED	Ò	D2
Ó	EE	Ó	D3
Õ	EF	Õ	D5
0	F0	0	30
1	F1	1	31
2	F2	2	32
3	F3	3	33

CP284SPAIN Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
4	F4	4	34
5	F5	5	35
6	F6	6	36
7	F7	7	37
8	F8	8	38
9	F9	9	39
³	FA	³	B3
Û	FB	Û	DB
Ü	FC	Ü	DC
Ù	FD	Ù	D9
Ú	FE	Ú	DA
APC	FF	APC	9F

CP285UK to ISO 8859-1 8-bit ASCII

The following table shows IBM Code Page 285(UK EBCDIC) to ISO 8859-1 translation.

CP285UK Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
SEL	04	ST	9C
HT	05	HT	09
RNL	06	SSA	86
DEL	07	DEL	7F
GE	08	EPA	97
SPS	09	RI	8D
RPT	0A	SS2	8E
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO	0E	SO/LS1	0E
SI	0F	SI/LS0	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
RES/ENP	14	OSC	9D
NL	15	NEL	85
BS	16	BS	08
POC	17	ESA	87

CP285UK Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
UBS	1A	PU2	92
CU1	1B	SS3	8F
IFS	1C	IFS	1C
IGS	1D	GS	1D
IRS	1E	RS	1E
IUS	1F	US	1F
DS	20	DS	80
SOS	21		81
FS	22	BPH	82
WUS	23	NBH	83
BYP/INP	24	IND	84
LF	25	LF	0A
ETB	26	ETB	17
ESC	27	ESC	1B
SA	28	HTS	88
SFE	29	HTJ	89
SM/SW	2A	VTS	8A
CSP	2B	PLD	8B
MFA	2C	PLU	8C
ENQ	2D	ENQ	05
ACK	2E	ACK	06
BEL	2F	BEL	07

CP285UK Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
	30	DCS	90
	31	PU1	91
	32	SYN	16
IR	33	STS	93
PP	34	CCH	94
TRN	35	MW	95
NBS	36	SPA	96
EOT	37	EOT	04
SBS	38	SOS	98
IT	39		99
RFF	3A	SCI	9A
CU3	3B	CSI	9B
DC4	3C	DC4	14
NAK	3D	NAK	15
	3E	PM	9E
SUB	3F	SUB	1A
Space	40	Space	20
NBSP	41	NBSP	A0
â	42	â	E2
ä	43	ä	E4
à	44	à	E0
á	45	á	E1
ã	46	ã	E3
å	47	å	E5
ç	48	ç	E7
ñ	49	ñ	F1
$	4A	$	24
.	4B	.	2E

CP285UK Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
<	4C	<	3C
(	4D	(	28
+	4E	+	2B
|	4F	|	7C
&	50	&	26
é	51	é	E9
ê	52	ê	EA
ë	53	ë	EB
è	54	è	E8
í	55	í	ED
î	56	î	EE
ï	57	ï	EF
ì	58	ì	EC
ß	59	ß	DF
!	5A	!	21
£	5B	£	A3
*	5C	*	2A
)	5D	)	29
;	5E	;	3B
¬	5F	¬	AC
-	60	-	2D
/	61	/	2F
Â	62	Â	C2
Ä	63	Ä	C4
À	64	À	C0
Á	65	Á	C1
Ã	66	Ã	C3
Å	67	Å	C5

CP285UK Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ç	68	Ç	C7
Ñ	69	Ñ	D1
¦	6A	¦	A6
,	6B	,	2C
%	6C	%	25
_	6D	_	5F
>	6E	>	3E
?	6F	?	3F
ø	70	ø	F8
É	71	É	C9
Ê	72	Ê	CA
Ë	73	Ë	CB
È	74	È	C8
Í	75	Í	CD
Î	76	Î	CE
Ï	77	Ï	CF
Ì	78	Ì	CC
`	79	`	60
:	7A	:	3A
#	7B	#	23
@	7C	@	40
'	7D	'	27
=	7E	=	3D
"	7F	"	22
Ø	80	Ø	D8
a	81	a	61
b	82	b	62
c	83	c	63

CP285UK Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
d	84	d	64
e	85	e	65
f	86	f	66
g	87	g	67
h	88	h	68
i	89	i	69
«	8A	«	AB
»	8B	»	BB
ð	8C	ð	F0
ý	8D	ý	FD
þ	8E	þ	FE
±	8F	±	B1
°	90	°	B0
j	91	j	6A
k	92	k	6B
l	93	l	6C
m	94	m	6D
n	95	n	6E
o	96	o	6F
p	97	p	70
q	98	q	71
r	99	r	72
ª	9A	ª	AA
º	9B	º	BA
æ	9C	æ	E6
¸	9D	¸	B8
Æ	9E	Æ	C6
¤	9F	¤	A4

CP285UK Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
µ	A0	µ	B5
_	A1	_	AF
s	A2	s	73
t	A3	t	74
u	A4	u	75
v	A5	v	76
w	A6	w	77
x	A7	x	78
y	A8	y	79
z	A9	z	7A
¡	AA	¡	A1
¿	AB	¿	BF
Ð	AC	Ð	D0
Ý	AD	Ý	DD
Þ	AE	Þ	DE
®	AF	®	AE
¢	B0	¢	A2
[	B1	[	5B
¥	B2	¥	A5
·	B3	·	B7
©	B4	©	A9
§	B5	§	A7
¶	B6	¶	B6
¼	B7	¼	BC
½	B8	½	BD
¾	B9	¾	BE
^	BA	^	5E
]	BB	]	5D

CP285UK Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
~	BC	~	7E
¨	BD	¨	A8
´	BE	´	B4
×	BF	×	D7
{	C0	{	7B
A	C1	A	41
B	C2	B	42
C	C3	C	43
D	C4	D	44
E	C5	E	45
F	C6	F	46
G	C7	G	47
H	C8	H	48
I	C9	I	49
-	CA	-	AD
ô	CB	ô	F4
ö	CC	ö	F6
ò	CD	ò	F2
ó	CE	ó	F3
õ	CF	õ	F5
}	D0	}	7D
J	D1	J	4A
K	D2	K	4B
L	D3	L	4C
M	D4	M	4D
N	D5	N	4E
O	D6	O	4F
P	D7	P	50

CP285UK Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
Q	D8	Q	51
R	D9	R	52
¹	DA	¹	B9
û	DB	û	FB
ü	DC	ü	FC
ù	DD	ù	F9
ú	DE	ú	FA
ÿ	DF	ÿ	FF
\	E0	\	5C
÷	E1	÷	F7
S	E2	S	53
T	E3	T	54
U	E4	U	55
V	E5	V	56
W	E6	W	57
X	E7	X	58
Y	E8	Y	59
Z	E9	Z	5A
²	EA	²	B2
Ô	EB	Ô	D4
Ö	EC	Ö	D6
Ò	ED	Ò	D2
Ó	EE	Ó	D3
Õ	EF	Õ	D5
0	F0	0	30
1	F1	1	31
2	F2	2	32
3	F3	3	33

CP285UK Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
4	F4	4	34
5	F5	5	35
6	F6	6	36
7	F7	7	37
8	F8	8	38
9	F9	9	39
³	FA	³	B3
Û	FB	Û	DB
Ü	FC	Ü	DC
Ù	FD	Ù	D9
Ú	FE	Ú	DA
APC	FF	APC	9F

CP871ICELAND to ISO 8859-1 8-bit ASCII

The following table shows IBM Code Page 871(Iceland EBCDIC) to ISO 8859-1 translation.

CP871ICELAND Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
SEL	04	ST	9C
HT	05	HT	09
RNL	06	SSA	86
DEL	07	DEL	7F
GE	08	EPA	97
SPS	09	RI	8D
RPT	0A	SS2	8E
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO	0E	SO/LS1	0E
SI	0F	SI/LS0	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
RES/ENP	14	OSC	9D
NL	15	NEL	85
BS	16	BS	08
POC	17	ESA	87
CAN	18	CAN	18

CP871ICELAND Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
EM	19	EM	19
UBS	1A	PU2	92
CU1	1B	SS3	8F
IFS	1C	IFS	1C
IGS	1D	GS	1D
IRS	1E	RS	1E
IUS	1F	US	1F
DS	20	DS	80
SOS	21		81
FS	22	BPH	82
WUS	23	NBH	83
BYP/INP	24	IND	84
LF	25	LF	0A
ETB	26	ETB	17
ESC	27	ESC	1B
SA	28	HTS	88
SFE	29	HTJ	89
SM/SW	2A	VTS	8A
CSP	2B	PLD	8B
MFA	2C	PLU	8C
ENQ	2D	ENQ	05
ACK	2E	ACK	06
BEL	2F	BEL	07
	30	DCS	90
	31	PU1	91

CP871ICELAND Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
	32	SYN	16
IR	33	STS	93
PP	34	CCH	94
TRN	35	MW	95
NBS	36	SPA	96
EOT	37	EOT	04
SBS	38	SOS	98
IT	39		99
RFF	3A	SCI	9A
CU3	3B	CSI	9B
DC4	3C	DC4	14
NAK	3D	NAK	15
	3E	PM	9E
SUB	3F	SUB	1A
Space	40	Space	20
NBSP	41	NBSP	A0
â	42	â	E2
ä	43	ä	E4
à	44	à	E0
á	45	á	E1
ã	46	ã	E3
å	47	å	E5
ç	48	ç	E7
ñ	49	ñ	F1
Þ	4A	Þ	DE
.	4B	.	2E
<	4C	<	3C
(	4D	(	28

CP871ICELAND Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
+	4E	+	2B
!	4F	!	21
&	50	&	26
é	51	é	E9
ê	52	ê	EA
ë	53	ë	EB
è	54	è	E8
í	55	í	ED
î	56	î	EE
ï	57	ï	EF
ì	58	ì	EC
ß	59	ß	DF
Æ	5A	Æ	C6
$	5B	$	24
*	5C	*	2A
)	5D	)	29
;	5E	;	3B
Ö	5F	Ö	D6
-	60	-	2D
/	61	/	2F
Â	62	Â	C2
Ä	63	Ä	C4
À	64	À	C0
Á	65	Á	C1
Ã	66	Ã	C3
Å	67	Å	C5
Ç	68	Ç	C7
Ñ	69	Ñ	D1

CP871ICELAND Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
¦	6A	¦	A6
,	6B	,	2C
%	6C	%	25
_	6D	_	5F
>	6E	>	3E
?	6F	?	3F
ø	70	ø	F8
É	71	É	C9
Ê	72	Ê	CA
Ë	73	Ë	CB
È	74	È	C8
Í	75	Í	CD
Î	76	Î	CE
Ï	77	Ï	CF
Ì	78	Ì	CC
ð	79	ð	F0
:	7A	:	3A
#	7B	#	23
Ð	7C	Ð	D0
'	7D	'	27
=	7E	=	3D
"	7F	"	22
Ø	80	Ø	D8
a	81	a	61
b	82	b	62
c	83	c	63
d	84	d	64
e	85	e	65

CP871ICELAND Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
f	86	f	66
g	87	g	67
h	88	h	68
i	89	i	69
«	8A	«	AB
»	8B	»	BB
`	8C	`	60
ý	8D	ý	FD
{	8E	{	7B
±	8F	±	B1
°	90	°	B0
j	91	j	6A
k	92	k	6B
l	93	l	6C
m	94	m	6D
n	95	n	6E
o	96	o	6F
p	97	p	70
q	98	q	71
r	99	r	72
ª	9A	ª	AA
º	9B	º	BA
}	9C	}	7D
¸	9D	¸	B8
]	9E	]	5D
¤	9F	¤	A4
µ	A0	µ	B5
ö	A1	ö	F6

CP871ICELAND Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
s	A2	s	73
t	A3	t	74
u	A4	u	75
v	A5	v	76
w	A6	w	77
x	A7	x	78
y	A8	y	79
z	A9	z	7A
¡	AA	¡	A1
¿	AB	¿	BF
@	AC	@	40
Ý	AD	Ý	DD
[	AE	[	5B
®	AF	®	AE
¢	B0	¢	A2
£	B1	£	A3
¥	B2	¥	A5
·	B3	·	B7
©	B4	©	A9
§	B5	§	A7
¶	B6	¶	B6
¼	B7	¼	BC
½	B8	½	BD
¾	B9	¾	BE
¬	BA	¬	AC
|	BB	|	7C
_	BC	_	AF
¨	BD	¨	A8

CP871ICELAND Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
\	BE	\	5C
×	BF	×	D7
þ	C0	þ	FE
A	C1	A	41
B	C2	B	42
C	C3	C	43
D	C4	D	44
E	C5	E	45
F	C6	F	46
G	C7	G	47
H	C8	H	48
I	C9	I	49
-	CA	-	AD
ô	CB	ô	F4
~	CC	~	7E
ò	CD	ò	F2
ó	CE	ó	F3
õ	CF	õ	F5
æ	D0	æ	E6
J	D1	J	4A
K	D2	K	4B
L	D3	L	4C
M	D4	M	4D
N	D5	N	4E
O	D6	O	4F
P	D7	P	50
Q	D8	Q	51
R	D9	R	52

CP871ICELAND Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
¹	DA	¹	B9
û	DB	û	FB
ü	DC	ü	FC
ù	DD	ù	F9
ú	DE	ú	FA
ÿ	DF	ÿ	FF
´	E0	´	B4
÷	E1	÷	F7
S	E2	S	53
T	E3	T	54
U	E4	U	55
V	E5	V	56
W	E6	W	57
X	E7	X	58
Y	E8	Y	59
Z	E9	Z	5A
²	EA	²	B2
Ô	EB	Ô	D4
^	EC	^	5E
Ò	ED	Ò	D2
Ó	EE	Ó	D3
Õ	EF	Õ	D5
0	F0	0	30
1	F1	1	31
2	F2	2	32
3	F3	3	33
4	F4	4	34
5	F5	5	35

CP871ICELAND Data		ISO 8859-1 Translation	
Control/Graphic	Hex Value	Character	Hex Value
6	F6	6	36
7	F7	7	37
8	F8	8	38
9	F9	9	39
³	FA	³	B3
Û	FB	Û	DB
Ü	FC	Ü	DC
Ù	FD	Ù	D9
Ú	FE	Ú	DA
APC	FF	APC	9F

ISO 8859-1 8-bit ASCII to CP37USA

The following table shows ISO 8859-1 8-bit ASCII to Code Page 37 translation.

ISO 8859-1 Data		CP37USA Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
EOT	04	EOT	37
ENQ	05	ENQ	2D
ACK	06	ACK	2E
BEL	07	BEL	2F
BS	08	BS	16
HT	09	HT	05
LF	0A	LF	25
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO/LS1	0E	SO	0E
SI/LS0	0F	SI	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
DC4	14	DC4	3C
NAK	15	NAK	3D
SYN	16	SYN	32
ETB	17	ETB	26
CAN	18	CAN	18

ISO 8859-1 Data		CP37USA Translation	
Control/Graphic	Hex Value	Character	Hex Value
EM	19	EM	19
SUB	1A	SUB	3F
ESC	1B	ESC	27
IFS	1C	IFS	1C
GS	1D	IGS	1D
RS	1E	IRS	1E
US	1F	IUS	1F
Space	20	Space	40
!	21	!	5A
"	22	"	7F
#	23	#	7B
$	24	$	5B
%	25	%	6C
&	26	&	50
'	27	'	7D
(	28	(	4D
)	29	)	5D
*	2A	*	5C
+	2B	+	4E
,	2C	,	6B
-	2D	-	60
.	2E	.	4B
/	2F	/	61
0	30	0	F0
1	31	1	F1

ISO 8859-1 Data		CP37USA Translation	
Control/Graphic	Hex Value	Character	Hex Value
2	32	2	F2
3	33	3	F3
4	34	4	F4
5	35	5	F5
6	36	6	F6
7	37	7	F7
8	38	8	F8
9	39	9	F9
:	3A	:	7A
;	3B	;	5E
<	3C	<	4C
=	3D	=	7E
>	3E	>	6E
?	3F	?	6F
@	40	@	7C
A	41	A	C1
B	42	B	C2
C	43	C	C3
D	44	D	C4
E	45	E	C5
F	46	F	C6
G	47	G	C7
H	48	H	C8
I	49	I	C9
J	4A	J	D1
K	4B	K	D2
L	4C	L	D3
M	4D	M	D4

ISO 8859-1 Data		CP37USA Translation	
Control/Graphic	Hex Value	Character	Hex Value
N	4E	N	D5
O	4F	O	D6
P	50	P	D7
Q	51	Q	D8
R	52	R	D9
S	53	S	E2
T	54	T	E3
U	55	U	E4
V	56	V	E5
W	57	W	E6
X	58	X	E7
Y	59	Y	E8
Z	5A	Z	E9
[	5B	[	BA
\	5C	\	E0
]	5D	]	BB
^	5E	^	B0
_	5F	_	6D
`	60	`	79
a	61	a	81
b	62	b	82
c	63	c	83
d	64	d	84
e	65	e	85
f	66	f	86
g	67	g	87
h	68	h	88
i	69	i	89

ISO 8859-1 Data		CP37USA Translation	
Control/Graphic	Hex Value	Character	Hex Value
j	6A	j	91
k	6B	k	92
l	6C	l	93
m	6D	m	94
n	6E	n	95
o	6F	o	96
p	70	p	97
q	71	q	98
r	72	r	99
s	73	s	A2
t	74	t	A3
u	75	u	A4
v	76	v	A5
w	77	w	A6
x	78	x	A7
y	79	y	A8
z	7A	z	A9
{	7B	{	C0
|	7C	|	4F
}	7D	}	D0
~	7E	~	A1
DEL	7F	DEL	07
	80	DS	20
	81	SOS	21
BPH	82	FS	22
NBH	83	WUS	23
IND	84	BYP/INP	24
NEL	85	NL	15

ISO 8859-1 Data		CP37USA Translation	
Control/Graphic	Hex Value	Character	Hex Value
SSA	86	RNL	06
ESA	87	POC	17
HTS	88	SA	28
HTJ	89	SFE	29
VTS	8A	SM/SW	2A
PLD	8B	CSP	2B
PLU	8C	MFA	2C
RI	8D	SPS	09
SS2	8E	RPT	0A
SS3	8F	CU1	1B
DCS	90		30
PU1	91		31
PU2	92	UBS	1A
STS	93	IR	33
CCH	94	PP	34
MW	95	TRN	35
SBA	96	NBS	36
EPA	97	GE	08
SOS	98	SBS	38
	99	IT	39
SCI	9A	RFF	3A
CSI	9B	CU3	3B
ST	9C	SEL	04
OSC	9D	RES/ENP	14
PM	9E		3E
APC	9F	EO	FF
NBSP	A0	NBSP	41
¡	A1	¡	AA

ISO 8859-1 Data		CP37USA Translation	
Control/Graphic	Hex Value	Character	Hex Value
¢	A2	¢	4A
£	A3	£	B1
¤	A4	¤	9F
¥	A5	¥	B2
¦	A6	¦	6A
§	A7	§	B5
¨	A8	¨	BD
©	A9	©	B4
ª	AA	ª	9A
«	AB	«	8A
¬	AC	¬	5F
-	AD	-	CA
®	AE	®	AF
_	AF	_	BC
°	B0	°	90
±	B1	±	8F
²	B2	²	EA
³	B3	³	FA
´	B4	´	BE
µ	B5	µ	A0
¶	B6	¶	B6
·	B7	·	B3
¸	B8	¸	9D
¹	B9	¹	DA
º	BA	º	9B
»	BB	»	8B
¼	BC	¼	B7
½	BD	½	B8

ISO 8859-1 Data		CP37USA Translation	
Control/Graphic	Hex Value	Character	Hex Value
¾	BE	¾	B9
¿	BF	¿	AB
À	C0	À	64
Á	C1	Á	65
Â	C2	Â	62
Ã	C3	Ã	66
Ä	C4	Ä	63
Å	C5	Å	67
Æ	C6	Æ	9E
Ç	C7	Ç	68
È	C8	È	74
É	C9	É	71
Ê	CA	Ê	72
Ë	CB	Ë	73
Ì	CC	Ì	78
Í	CD	Í	75
Î	CE	Î	76
Ï	CF	Ï	77
Ð	D0	Ð	AC
Ñ	D1	Ñ	69
Ò	D2	Ò	ED
Ó	D3	Ó	EE
Ô	D4	Ô	EB
Õ	D5	Õ	EF
Ö	D6	Ö	EC
×	D7	×	BF
Ø	D8	Ø	80
Ù	D9	Ù	FD

ISO 8859-1 Data		CP37USA Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ú	DA	Ú	FE
Û	DB	Û	FB
Ü	DC	Ü	FC
Ý	DD	Ý	AD
Þ	DE	Þ	AE
ß	DF	ß	59
à	E0	à	44
á	E1	á	45
â	E2	â	42
ã	E3	ã	46
ä	E4	ä	43
å	E5	å	47
æ	E6	æ	9C
ç	E7	ç	48
è	E8	è	54
é	E9	é	51
ê	EA	ê	52
ë	EB	ë	53
ì	EC	ì	58
í	ED	í	55
î	EE	î	56
ï	EF	ï	57
ð	F0	ð	8C
ñ	F1	ñ	49
ò	F2	ò	CD
ó	F3	ó	CE
ô	F4	ô	CB
õ	F5	õ	CF

ISO 8859-1 Data		CP37USA Translation	
Control/Graphic	Hex Value	Character	Hex Value
ö	F6	ö	CC
÷	F7	÷	E1
ø	F8	ø	70
ù	F9	ù	DD
ú	FA	ú	DE
û	FB	û	DB
ü	FC	ü	DC
ý	FD	ý	8D
þ	FE	þ	8E
ÿ	FF	ÿ	DF

ISO 8859-1 8-bit ASCII to CP297FRANCE

The following table shows ISO 8859-1 to IBM Code Page 297(French EBCDIC) translation.

ISO 8859-1 Data		CP297FRANCE Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
EOT	04	EOT	37
ENQ	05	ENQ	2D
ACK	06	ACK	2E
BEL	07	BEL	2F
BS	08	BS	16
HT	09	HT	05
LF	0A	LF	25
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO/LS1	0E	SO	0E
SI/LS0	0F	SI	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
DC4	14	DC4	3C
NAK	15	NAK	3D
SYN	16	SYN	32
ETB	17	ETB	26
CAN	18	CAN	18

ISO 8859-1 Data		CP297FRANCE Translation	
Control/Graphic	Hex Value	Character	Hex Value
EM	19	EM	19
SUB	1A	SUB	3F
ESC	1B	ESC	27
IFS	1C	IFS	1C
GS	1D	IGS	1D
RS	1E	IRS	1E
US	1F	IUS	1F
Space	20	Space	40
!	21	!	4F
"	22	"	7F
#	23	#	B1
$	24	$	5B
%	25	%	6C
&	26	&	50
'	27	'	7D
(	28	(	4D
)	29	)	5D
*	2A	*	5C
+	2B	+	4E
,	2C	,	6B
-	2D	-	60
.	2E	.	4B
/	2F	/	61
0	30	0	F0
1	31	1	F1

ISO 8859-1 Data		CP297FRANCE Translation	
Control/Graphic	Hex Value	Character	Hex Value
2	32	2	F2
3	33	3	F3
4	34	4	F4
5	35	5	F5
6	36	6	F6
7	37	7	F7
8	38	8	F8
9	39	9	F9
:	3A	:	7A
;	3B	;	5E
<	3C	<	4C
=	3D	=	7E
>	3E	>	6E
?	3F	?	6F
@	40	@	44
A	41	A	C1
B	42	B	C2
C	43	C	C3
D	44	D	C4
E	45	E	C5
F	46	F	C6
G	47	G	C7
H	48	H	C8
I	49	I	C9
J	4A	J	D1
K	4B	K	D2
L	4C	L	D3
M	4D	M	D4

ISO 8859-1 Data		CP297FRANCE Translation	
Control/Graphic	Hex Value	Character	Hex Value
N	4E	N	D5
O	4F	O	D6
P	50	P	D7
Q	51	Q	D8
R	52	R	D9
S	53	S	E2
T	54	T	E3
U	55	U	E4
V	56	V	E5
W	57	W	E6
X	58	X	E7
Y	59	Y	E8
Z	5A	Z	E9
[	5B	[	90
\	5C	\	48
]	5D	]	B5
^	5E	^	5F
_	5F	_	6D
`	60	`	A0
a	61	a	81
b	62	b	82
c	63	c	83
d	64	d	84
e	65	e	85
f	66	f	86
g	67	g	87
h	68	h	88
i	69	i	89

ISO 8859-1 Data		CP297FRANCE Translation	
Control/Graphic	Hex Value	Character	Hex Value
j	6A	j	91
k	6B	k	92
l	6C	l	93
m	6D	m	94
n	6E	n	95
o	6F	o	96
p	70	p	97
q	71	q	98
r	72	r	99
s	73	s	A2
t	74	t	A3
u	75	u	A4
v	76	v	A5
w	77	w	A6
x	78	x	A7
y	79	y	A8
z	7A	z	A9
{	7B	{	51
|	7C	|	BB
}	7D	}	54
~	7E	~	BD
DEL	7F	DEL	07
	80	DS	20
	81	SOS	21
BPH	82	FS	22
NBH	83	WUS	23
IND	84	BYP/INP	24
NEL	85	NL	15

ISO 8859-1 Data		CP297FRANCE Translation	
Control/Graphic	Hex Value	Character	Hex Value
SSA	86	RNL	06
ESA	87	POC	17
HTS	88	SA	28
HTJ	89	SFE	29
VTS	8A	SM/SW	2A
PLD	8B	CSP	2B
PLU	8C	MFA	2C
RI	8D	SPS	09
SS2	8E	RPT	0A
SS3	8F	CU1	1B
DCS	90		30
PU1	91		31
PU2	92	UBS	1A
STS	93	IR	33
CCH	94	PP	34
MW	95	TRN	35
SBA	96	NBS	36
EPA	97	GE	08
SOS	98	SBS	38
	99	IT	39
SCI	9A	RFF	3A
CSI	9B	CU3	3B
ST	9C	SEL	04
OSC	9D	RES/ENP	14
PM	9E		3E
APC	9F	EO	FF
NBSP	A0	NBSP	41
¡	A1	¡	AA

ISO 8859-1 Data		CP297FRANCE Translation	
Control/Graphic	Hex Value	Character	Hex Value
¢	A2	¢	B0
£	A3	£	7B
¤	A4	¤	9F
¥	A5	¥	B2
¦	A6	¦	DD
§	A7	§	5A
¨	A8	¨	A1
©	A9	©	B4
ª	AA	ª	9A
«	AB	«	8A
¬	AC	¬	BA
-	AD	-	CA
®	AE	®	AF
_	AF	_	BC
°	B0	°	4A
±	B1	±	8F
²	B2	²	EA
³	B3	³	FA
´	B4	´	BE
µ	B5	µ	79
¶	B6	¶	B6
·	B7	·	B3
¸	B8	¸	9D
¹	B9	¹	DA
º	BA	º	9B
»	BB	»	8B
¼	BC	¼	B7
½	BD	½	B8

ISO 8859-1 Data		CP297FRANCE Translation	
Control/Graphic	Hex Value	Character	Hex Value
¾	BE	¾	B9
¿	BF	¿	AB
À	C0	À	64
Á	C1	Á	65
Â	C2	Â	62
Ã	C3	Ã	66
Ä	C4	Ä	63
Å	C5	Å	67
Æ	C6	Æ	9E
Ç	C7	Ç	68
È	C8	È	74
É	C9	É	71
Ê	CA	Ê	72
Ë	CB	Ë	73
Ì	CC	Ì	78
Í	CD	Í	75
Î	CE	Î	76
Ï	CF	Ï	77
Ð	D0	Ð	AC
Ñ	D1	Ñ	69
Ò	D2	Ò	ED
Ó	D3	Ó	EE
Ô	D4	Ô	EB
Õ	D5	Õ	EF
Ö	D6	Ö	EC
×	D7	×	BF
Ø	D8	Ø	80
Ù	D9	Ù	FD

ISO 8859-1 Data		CP297FRANCE Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ú	DA	Ú	FE
Û	DB	Û	FB
Ü	DC	Ü	FC
Ý	DD	Ý	AD
Þ	DE	Þ	AE
ß	DF	ß	59
à	E0	à	7C
á	E1	á	45
â	E2	â	42
ã	E3	ã	46
ä	E4	ä	43
å	E5	å	47
æ	E6	æ	9C
ç	E7	ç	E0
è	E8	è	D0
é	E9	é	C0
ê	EA	ê	52
ë	EB	ë	53
ì	EC	ì	58
í	ED	í	55
î	EE	î	56
ï	EF	ï	57
ð	F0	ð	8C
ñ	F1	ñ	49
ò	F2	ò	CD
ó	F3	ó	CE
ô	F4	ô	CB
õ	F5	õ	CF

ISO 8859-1 Data		CP297FRANCE Translation	
Control/Graphic	Hex Value	Character	Hex Value
ö	F6	ö	CC
÷	F7	÷	E1
ø	F8	ø	70
ù	F9	ù	6A
ú	FA	ú	DE
û	FB	û	DB
ü	FC	ü	DC
ý	FD	ý	8D
þ	FE	þ	8E
ÿ	FF	ÿ	DF

ISO 8859-1 8-bit ASCII to CP500GLOBAL

The following table shows ISO 8859-1 to IBM Code Page 500(Global EBCDIC) translation.

ISO 8859-1 Data		CP500GLOBAL Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
EOT	04	EOT	37
ENQ	05	ENQ	2D
ACK	06	ACK	2E
BEL	07	BEL	2F
BS	08	BS	16
HT	09	HT	05
LF	0A	LF	25
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO/LS1	0E	SO	0E
SI/LS0	0F	SI	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
DC4	14	DC4	3C
NAK	15	NAK	3D
SYN	16	SYN	32
ETB	17	ETB	26
CAN	18	CAN	18

ISO 8859-1 Data		CP500GLOBAL Translation	
Control/Graphic	Hex Value	Character	Hex Value
EM	19	EM	19
SUB	1A	SUB	3F
ESC	1B	ESC	27
IFS	1C	IFS	1C
GS	1D	IGS	1D
RS	1E	IRS	1E
US	1F	IUS	1F
Space	20	Space	40
!	21	!	4F
"	22	"	7F
#	23	#	7B
$	24	$	5B
%	25	%	6C
&	26	&	50
'	27	'	7D
(	28	(	4D
)	29	)	5D
*	2A	*	5C
+	2B	+	4E
,	2C	,	6B
-	2D	-	60
.	2E	.	4B
/	2F	/	61
0	30	0	F0
1	31	1	F1

ISO 8859-1 Data		CP500GLOBAL Translation	
Control/Graphic	Hex Value	Character	Hex Value
2	32	2	F2
3	33	3	F3
4	34	4	F4
5	35	5	F5
6	36	6	F6
7	37	7	F7
8	38	8	F8
9	39	9	F9
:	3A	:	7A
;	3B	;	5E
<	3C	<	4C
=	3D	=	7E
>	3E	>	6E
?	3F	?	6F
@	40	@	7C
A	41	A	C1
B	42	B	C2
C	43	C	C3
D	44	D	C4
E	45	E	C5
F	46	F	C6
G	47	G	C7
H	48	H	C8
I	49	I	C9
J	4A	J	D1
K	4B	K	D2
L	4C	L	D3
M	4D	M	D4

ISO 8859-1 Data		CP500GLOBAL Translation	
Control/Graphic	Hex Value	Character	Hex Value
N	4E	N	D5
O	4F	O	D6
P	50	P	D7
Q	51	Q	D8
R	52	R	D9
S	53	S	E2
T	54	T	E3
U	55	U	E4
V	56	V	E5
W	57	W	E6
X	58	X	E7
Y	59	Y	E8
Z	5A	Z	E9
[	5B	[	4A
\	5C	\	E0
]	5D	]	5A
^	5E	^	5F
_	5F	_	6D
`	60	`	79
a	61	a	81
b	62	b	82
c	63	c	83
d	64	d	84
e	65	e	85
f	66	f	86
g	67	g	87
h	68	h	88
i	69	i	89

ISO 8859-1 Data		CP500GLOBAL Translation	
Control/Graphic	Hex Value	Character	Hex Value
j	6A	j	91
k	6B	k	92
l	6C	l	93
m	6D	m	94
n	6E	n	95
o	6F	o	96
p	70	p	97
q	71	q	98
r	72	r	99
s	73	s	A2
t	74	t	A3
u	75	u	A4
v	76	v	A5
w	77	w	A6
x	78	x	A7
y	79	y	A8
z	7A	z	A9
{	7B	{	C0
|	7C	|	BB
}	7D	}	D0
~	7E	~	A1
DEL	7F	DEL	07
	80	DS	20
	81	SOS	21
BPH	82	FS	22
NBH	83	WUS	23
IND	84	BYP/INP	24
NEL	85	NL	15

ISO 8859-1 Data		CP500GLOBAL Translation	
Control/Graphic	Hex Value	Character	Hex Value
SSA	86	RNL	06
ESA	87	POC	17
HTS	88	SA	28
HTJ	89	SFE	29
VTS	8A	SM/SW	2A
PLD	8B	CSP	2B
PLU	8C	MFA	2C
RI	8D	SPS	09
SS2	8E	RPT	0A
SS3	8F	CU1	1B
DCS	90		30
PU1	91		31
PU2	92	UBS	1A
STS	93	IR	33
CCH	94	PP	34
MW	95	TRN	35
SBA	96	NBS	36
EPA	97	GE	08
SOS	98	SBS	38
	99	IT	39
SCI	9A	RFF	3A
CSI	9B	CU3	3B
ST	9C	SEL	04
OSC	9D	RES/ENP	14
PM	9E		3E
APC	9F	EO	FF
NBSP	A0	NBSP	41
¡	A1	¡	AA

ISO 8859-1 Data		CP500GLOBAL Translation	
Control/Graphic	Hex Value	Character	Hex Value
¢	A2	¢	B0
£	A3	£	B1
¤	A4	¤	9F
¥	A5	¥	B2
¦	A6	¦	6A
§	A7	§	B5
¨	A8	¨	BD
©	A9	©	B4
ª	AA	ª	9A
«	AB	«	8A
¬	AC	¬	BA
-	AD	-	CA
®	AE	®	AF
¯	AF	¯	BC
°	B0	°	90
±	B1	±	8F
²	B2	²	EA
³	B3	³	FA
´	B4	´	BE
µ	B5	µ	A0
¶	B6	¶	B6
·	B7	·	B3
¸	B8	¸	9D
¹	B9	¹	DA
º	BA	º	9B
»	BB	»	8B
¼	BC	¼	B7
½	BD	½	B8

ISO 8859-1 Data		CP500GLOBAL Translation	
Control/Graphic	Hex Value	Character	Hex Value
¾	BE	¾	B9
¿	BF	¿	AB
À	C0	À	64
Á	C1	Á	65
Â	C2	Â	62
Ã	C3	Ã	66
Ä	C4	Ä	63
Å	C5	Å	67
Æ	C6	Æ	9E
Ç	C7	Ç	68
È	C8	È	74
É	C9	É	71
Ê	CA	Ê	72
Ë	CB	Ë	73
Ì	CC	Ì	78
Í	CD	Í	75
Î	CE	Î	76
Ï	CF	Ï	77
Ð	D0	Ð	AC
Ñ	D1	Ñ	69
Ò	D2	Ò	ED
Ó	D3	Ó	EE
Ô	D4	Ô	EB
Õ	D5	Õ	EF
Ö	D6	Ö	EC
×	D7	×	BF
Ø	D8	Ø	80
Ù	D9	Ù	FD

ISO 8859-1 Data		CP500GLOBAL Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ú	DA	Ú	FE
Û	DB	Û	FB
Ü	DC	Ü	FC
Ý	DD	Ý	AD
Þ	DE	Þ	AE
ß	DF	ß	59
à	E0	à	44
á	E1	á	45
â	E2	â	42
ã	E3	ã	46
ä	E4	ä	43
å	E5	å	47
æ	E6	æ	9C
ç	E7	ç	48
è	E8	è	54
é	E9	é	51
ê	EA	ê	52
ë	EB	ë	53
ì	EC	ì	58
í	ED	í	55
î	EE	î	56
ï	EF	ï	57
ð	F0	ð	8C
ñ	F1	ñ	49
ò	F2	ò	CD
ó	F3	ó	CE
ô	F4	ô	CB
õ	F5	õ	CF

ISO 8859-1 Data		CP500GLOBAL Translation	
Control/Graphic	Hex Value	Character	Hex Value
ö	F6	ö	CC
÷	F7	÷	E1
ø	F8	ø	70
ù	F9	ù	DD
ú	FA	ú	DE
û	FB	û	DB
ü	FC	ü	DC
ý	FD	ý	8D
þ	FE	þ	8E
ÿ	FF	ÿ	DF

ISO 8859-1 8-bit ASCII to CP273GERMANY

The following table shows ISO 8859-1 8-bit ASCII to Code Page 273 translation.

ISO 8859-1 Data		CP273GERMANY Translation	
Control/ Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
EOT	04	EOT	37
ENQ	05	ENQ	2D
ACK	06	ACK	2E
BEL	07	BEL	2F
BS	08	BS	16
HT	09	HT	05
LF	0A	LF	25
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO/LS1	0E	SO	0E
SI/LS0	0F	SI	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
DC4	14	DC4	3C
NAK	15	NAK	3D
SYN	16	SYN	32
ETB	17	ETB	26

ISO 8859-1 Data		CP273GERMANY Translation	
Control/ Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
SUB	1A	SUB	3F
ESC	1B	ESC	27
IFS	1C	IFS	1C
GS	1D	IGS	1D
RS	1E	IRS	1E
US	1F	IUS	1F
Space	20	Space	40
!	21	!	4F
"	22	"	7F
#	23	#	7B
$	24	$	5B
%	25	%	6C
&	26	&	50
'	27	'	7D
(	28	(	4D
)	29	)	5D
*	2A	*	5C
+	2B	+	4E
,	2C	,	6B
-	2D	-	60
.	2E	.	4B
/	2F	/	61

ISO 8859-1 Data		CP273GERMANY Translation	
Control/ Graphic	Hex Value	Character	Hex Value
0	30	0	F0
1	31	1	F1
2	32	2	F2
3	33	3	F3
4	34	4	F4
5	35	5	F5
6	36	6	F6
7	37	7	F7
8	38	8	F8
9	39	9	F9
:	3A	:	7A
;	3B	;	5E
<	3C	<	4C
=	3D	=	7E
>	3E	>	6E
?	3F	?	6F
@	40	@	B5
A	41	A	C1
B	42	B	C2
C	43	C	C3
D	44	D	C4
E	45	E	C5
F	46	F	C6
G	47	G	C7
H	48	H	C8
I	49	I	C9
J	4A	J	D1

ISO 8859-1 Data		CP273GERMANY Translation	
Control/ Graphic	Hex Value	Character	Hex Value
K	4B	K	D2
L	4C	L	D3
M	4D	M	D4
N	4E	N	D5
O	4F	O	D6
P	50	P	D7
Q	51	Q	D8
R	52	R	D9
S	53	S	E2
T	54	T	E3
U	55	U	E4
V	56	V	E5
W	57	W	E6
X	58	X	E7
Y	59	Y	E8
Z	5A	Z	E9
[	5B	[	63
\	5C	\	EC
]	5D	]	FC
^	5E	^	5F
_	5F	_	6D
`	60	`	79
a	61	a	81
b	62	b	82
c	63	c	83
d	64	d	84
e	65	e	85

ISO 8859-1 Data		CP273GERMANY Translation	
Control/ Graphic	Hex Value	Character	Hex Value
f	66	f	86
g	67	g	87
h	68	h	88
i	69	i	89
j	6A	j	91
k	6B	k	92
l	6C	l	93
m	6D	m	94
n	6E	n	95
o	6F	o	96
p	70	p	97
q	71	q	98
r	72	r	99
s	73	s	A2
t	74	t	A3
u	75	u	A4
v	76	v	A5
w	77	w	A6
x	78	x	A7
y	79	y	A8
z	7A	z	A9
{	7B	{	43
|	7C	|	BB
}	7D	}	DC
~	7E	~	59
DEL	7F	DEL	07
	80	DS	20
	81	SOS	21
BPH	82	FS	22
NBH	83	WUS	23
IND	84	BYP/INP	24
NEL	85	NL	15
SSA	86	RNL	06
ESA	87	POC	17
HTS	88	SA	28
HTJ	89	SFE	29
VTS	8A	SM/SW	2A
PLD	8B	CSP	2B
PLU	8C	MFA	2C
RI	8D	SPS	09
SS2	8E	RPT	0A
SS3	8F	CU1	1B
DCS	90		30
PU1	91		31
PU2	92	UBS	1A
STS	93	IR	33
CCH	94	PP	34
MW	95	TRN	35
SBA	96	NBS	36
EPA	97	GE	08
SOS	98	SBS	38
	99	IT	39
SCI	9A	RFF	3A
CSI	9B	CU3	3B

ISO 8859-1 Data		CP273GERMANY Translation	
Control/ Graphic	Hex Value	Character	Hex Value
ST	9C	SEL	04
OSC	9D	RES/ENP	14
PM	9E		3E
APC	9F	EO	FF
NBSP	A0	NBSP	41
¡	A1	¡	AA
¢	A2	¢	B0
£	A3	£	B1
¤	A4	¤	9F
¥	A5	¥	B2
¦	A6	¦	CC
§	A7	§	7C
¨	A8	¨	BD
©	A9	©	B4
ª	AA	ª	9A
«	AB	«	8A
¬	AC	¬	BA
-	AD	-	CA
®	AE	®	AF
¯	AF	¯	BC
°	B0	°	90
±	B1	±	8F
²	B2	²	EA
³	B3	³	FA
´	B4	´	BE
µ	B5	µ	A0
¶	B6	¶	B6

ISO 8859-1 Data		CP273GERMANY Translation	
Control/ Graphic	Hex Value	Character	Hex Value
·	B7	·	B3
¸	B8	¸	9D
¹	B9	¹	DA
º	BA	º	9B
»	BB	»	8B
¼	BC	¼	B7
½	BD	½	B8
¾	BE	¾	B9
¿	BF	¿	AB
À	C0	À	64
Á	C1	Á	65
Â	C2	Â	62
Ã	C3	Ã	66
Ä	C4	Ä	4A
Å	C5	Å	67
Æ	C6	Æ	9E
Ç	C7	Ç	68
È	C8	È	74
É	C9	É	71
Ê	CA	Ê	72
Ë	CB	Ë	73
Ì	CC	Ì	78
Í	CD	Í	75
Î	CE	Î	76
Ï	CF	Ï	77
Ð	D0	Ð	AC
Ñ	D1	Ñ	69

ISO 8859-1 Data		CP273GERMANY Translation	
Control/ Graphic	Hex Value	Character	Hex Value
Ò	D2	Ò	ED
Ó	D3	Ó	EE
Ô	D4	Ô	EB
Õ	D5	Õ	EF
Ö	D6	Ö	E0
×	D7	×	BF
Ø	D8	Ø	80
Ù	D9	Ù	FD
Ú	DA	Ú	FE
Û	DB	Û	FB
Ü	DC	Ü	5A
Ý	DD	Ý	AD
Þ	DE	Þ	AE
ß	DF	ß	A1
à	E0	à	44
á	E1	á	45
â	E2	â	42
ã	E3	ã	46
ä	E4	ä	C0
å	E5	å	47
æ	E6	æ	9C
ç	E7	ç	48
è	E8	è	54
é	E9	é	51
ê	EA	ê	52
ë	EB	ë	53
ì	EC	ì	58

ISO 8859-1 Data		CP273GERMANY Translation	
Control/ Graphic	Hex Value	Character	Hex Value
í	ED	í	55
î	EE	î	56
ï	EF	ï	57
ð	F0	ð	8C
ñ	F1	ñ	49
ò	F2	ò	CD
ó	F3	ó	CE
ô	F4	ô	CB
õ	F5	õ	CF
ö	F6	ö	6A
÷	F7	÷	E1
ø	F8	ø	70
ù	F9	ù	DD
ú	FA	ú	DE
û	FB	û	DB
ü	FC	ü	D0
ý	FD	ý	8D
þ	FE	þ	8E
ÿ	FF	ÿ	DF

ISO 8859-1 8-bit ASCII to CP277DENMARK

The following table shows ISO 8859-1 8-bit ASCII to Code Page 277 translation.

ISO 8859-1 Data		CP277DENMARK Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
EOT	04	EOT	37
ENQ	05	ENQ	2D
ACK	06	ACK	2E
BEL	07	BEL	2F
BS	08	BS	16
HT	09	HT	05
LF	0A	LF	25
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO/LS1	0E	SO	0E
SI/LS0	0F	SI	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
DC4	14	DC4	3C
NAK	15	NAK	3D
SYN	16	SYN	32
ETB	17	ETB	26

ISO 8859-1 Data		CP277DENMARK Translation	
Control/Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
SUB	1A	SUB	3F
ESC	1B	ESC	27
IFS	1C	IFS	1C
GS	1D	IGS	1D
RS	1E	IRS	1E
US	1F	IUS	1F
Space	20	Space	40
!	21	!	4F
"	22	"	7F
#	23	#	4A
$	24	$	67
%	25	%	6C
&	26	&	50
'	27	'	7D
(	28	(	4D
)	29	)	5D
*	2A	*	5C
+	2B	+	4E
,	2C	,	6B
-	2D	-	60
.	2E	.	4B
/	2F	/	61

ISO 8859-1 Data		CP277DENMARK Translation	
Control/Graphic	Hex Value	Character	Hex Value
0	30	0	F0
1	31	1	F1
2	32	2	F2
3	33	3	F3
4	34	4	F4
5	35	5	F5
6	36	6	F6
7	37	7	F7
8	38	8	F8
9	39	9	F9
:	3A	:	7A
;	3B	;	5E
<	3C	<	4C
=	3D	=	7E
>	3E	>	6E
?	3F	?	6F
@	40	@	80
A	41	A	C1
B	42	B	C2
C	43	C	C3
D	44	D	C4
E	45	E	C5
F	46	F	C6
G	47	G	C7
H	48	H	C8
I	49	I	C9
J	4A	J	D1

ISO 8859-1 Data		CP277DENMARK Translation	
Control/Graphic	Hex Value	Character	Hex Value
K	4B	K	D2
L	4C	L	D3
M	4D	M	D4
N	4E	N	D5
O	4F	O	D6
P	50	P	D7
Q	51	Q	D8
R	52	R	D9
S	53	S	E2
T	54	T	E3
U	55	U	E4
V	56	V	E5
W	57	W	E6
X	58	X	E7
Y	59	Y	E8
Z	5A	Z	E9
[	5B	[	9E
\	5C	\	E0
]	5D	]	9F
^	5E	^	5F
_	5F	_	6D
`	60	`	79
a	61	a	81
b	62	b	82
c	63	c	83
d	64	d	84
e	65	e	85

ISO 8859-1 Data		CP277DENMARK Translation	
Control/Graphic	Hex Value	Character	Hex Value
f	66	f	86
g	67	g	87
h	68	h	88
i	69	i	89
j	6A	j	91
k	6B	k	92
l	6C	l	93
m	6D	m	94
n	6E	n	95
o	6F	o	96
p	70	p	97
q	71	q	98
r	72	r	99
s	73	s	A2
t	74	t	A3
u	75	u	A4
v	76	v	A5
w	77	w	A6
x	78	x	A7
y	79	y	A8
z	7A	z	A9
{	7B	{	9C
|	7C	|	BB
}	7D	}	47
~	7E	~	DC
DEL	7F	DEL	07
	80	DS	20
	81	SOS	21
BPH	82	FS	22
NBH	83	WUS	23
IND	84	BYP/INP	24
NEL	85	NL	15
SSA	86	RNL	06
ESA	87	POC	17
HTS	88	SA	28
HTJ	89	SFE	29
VTS	8A	SM/SW	2A
PLD	8B	CSP	2B
PLU	8C	MFA	2C
RI	8D	SPS	09
SS2	8E	RPT	0A
SS3	8F	CU1	1B
DCS	90		30
PU1	91		31
PU2	92	UBS	1A
STS	93	IR	33
CCH	94	PP	34
MW	95	TRN	35
SBA	96	NBS	36
EPA	97	GE	08
SOS	98	SBS	38
	99	IT	39
SCI	9A	RFF	3A
CSI	9B	CU3	3B

ISO 8859-1 Data		CP277DENMARK Translation	
Control/Graphic	Hex Value	Character	Hex Value
ST	9C	SEL	04
OSC	9D	RES/ENP	14
PM	9E		3E
APC	9F	EO	FF
NBSP	A0	NBSP	41
¡	A1	¡	AA
¢	A2	¢	B0
£	A3	£	B1
¤	A4	¤	5A
¥	A5	¥	B2
¦	A6	¦	70
§	A7	§	B5
¨	A8	¨	BD
©	A9	©	B4
ª	AA	ª	9A
«	AB	«	8A
¬	AC	¬	BA
-	AD	-	CA
®	AE	®	AF
¯	AF	¯	BC
°	B0	°	90
±	B1	±	8F
²	B2	²	EA
³	B3	³	FA
´	B4	´	BE
µ	B5	µ	A0
¶	B6	¶	B6

ISO 8859-1 Data		CP277DENMARK Translation	
Control/Graphic	Hex Value	Character	Hex Value
·	B7	·	B3
¸	B8	¸	9D
¹	B9	¹	DA
º	BA	º	9B
»	BB	»	8B
¼	BC	¼	B7
½	BD	½	B8
¾	BE	¾	B9
¿	BF	¿	AB
À	C0	À	64
Á	C1	Á	65
Â	C2	Â	62
Ã	C3	Ã	66
Ä	C4	Ä	63
Å	C5	Å	5B
Æ	C6	Æ	7B
Ç	C7	Ç	68
È	C8	È	74
É	C9	É	71
Ê	CA	Ê	72
Ë	CB	Ë	73
Ì	CC	Ì	78
Í	CD	Í	75
Î	CE	Î	76
Ï	CF	Ï	77
Ð	D0	Ð	AC
Ñ	D1	Ñ	69

ISO 8859-1 Data		CP277DENMARK Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ò	D2	Ò	ED
Ó	D3	Ó	EE
Ô	D4	Ô	EB
Õ	D5	Õ	EF
Ö	D6	Ö	EC
×	D7	×	BF
Ø	D8	Ø	7C
Ù	D9	Ù	FD
Ú	DA	Ú	FE
Û	DB	Û	FB
Ü	DC	Ü	FC
Ý	DD	Ý	AD
Þ	DE	Þ	AE
ß	DF	ß	59
à	E0	à	44
á	E1	á	45
â	E2	â	42
ã	E3	ã	46
ä	E4	ä	43
å	E5	å	D0
æ	E6	æ	C0
ç	E7	ç	48
è	E8	è	54
é	E9	é	51
ê	EA	ê	52
ë	EB	ë	53
ì	EC	ì	58

ISO 8859-1 Data		CP277DENMARK Translation	
Control/Graphic	Hex Value	Character	Hex Value
í	ED	í	55
î	EE	î	56
ï	EF	ï	57
ð	F0	ð	8C
ñ	F1	ñ	49
ò	F2	ò	CD
ó	F3	ó	CE
ô	F4	ô	CB
õ	F5	õ	CF
ö	F6	ö	CC
÷	F7	÷	E1
ø	F8	ø	6A
ù	F9	ù	DD
ú	FA	ú	DE
û	FB	û	DB
ü	FC	ü	A1
ý	FD	ý	8D
þ	FE	þ	8E
ÿ	FF	ÿ	DF

ISO 8859-1 8-bit ASCII to CP278SWEDEN

The following table shows ISO 8859-1 8-bit ASCII to Code Page 278 translation.

ISO 8859-1 Data		CP278SWEDEN Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
EOT	04	EOT	37
ENQ	05	ENQ	2D
ACK	06	ACK	2E
BEL	07	BEL	2F
BS	08	BS	16
HT	09	HT	05
LF	0A	LF	25
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO/LS1	0E	SO	0E
SI/LS0	0F	SI	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
DC4	14	DC4	3C
NAK	15	NAK	3D
SYN	16	SYN	32
ETB	17	ETB	26
CAN	18	CAN	18
EM	19	EM	19
SUB	1A	SUB	3F
ESC	1B	ESC	27
IFS	1C	IFS	1C
GS	1D	IGS	1D
RS	1E	IRS	1E
US	1F	IUS	1F
Space	20	Space	40
!	21	!	4F
"	22	"	7F
#	23	#	63
$	24	$	67
%	25	%	6C
&	26	&	50
'	27	'	7D
(	28	(	4D
)	29	)	5D
*	2A	*	5C
+	2B	+	4E
,	2C	,	6B
-	2D	-	60
.	2E	.	4B
/	2F	/	61

ISO 8859-1 Data		CP278SWEDEN Translation	
Control/Graphic	Hex Value	Character	Hex Value
0	30	0	F0
1	31	1	F1
2	32	2	F2
3	33	3	F3
4	34	4	F4
5	35	5	F5
6	36	6	F6
7	37	7	F7
8	38	8	F8
9	39	9	F9
:	3A	:	7A
;	3B	;	5E
<	3C	<	4C
=	3D	=	7E
>	3E	>	6E
?	3F	?	6F
@	40	@	EC
A	41	A	C1
B	42	B	C2
C	43	C	C3
D	44	D	C4
E	45	E	C5
F	46	F	C6
G	47	G	C7
H	48	H	C8
I	49	I	C9
J	4A	J	D1
K	4B	K	D2

ISO 8859-1 Data		CP278SWEDEN Translation	
Control/Graphic	Hex Value	Character	Hex Value
L	4C	L	D3
M	4D	M	D4
N	4E	N	D5
O	4F	O	D6
P	50	P	D7
Q	51	Q	D8
R	52	R	D9
S	53	S	E2
T	54	T	E3
U	55	U	E4
V	56	V	E5
W	57	W	E6
X	58	X	E7
Y	59	Y	E8
Z	5A	Z	E9
[	5B	[	B5
\	5C	\	71
]	5D	]	9F
^	5E	^	5F
_	5F	_	6D
`	60	`	51
a	61	a	81
b	62	b	82
c	63	c	83
d	64	d	84
e	65	e	85
f	66	f	86
g	67	g	87

ISO 8859-1 Data		CP278SWEDEN Translation	
Control/Graphic	Hex Value	Character	Hex Value
h	68	h	88
i	69	i	89
j	6A	j	91
k	6B	k	92
l	6C	l	93
m	6D	m	94
n	6E	n	95
o	6F	o	96
p	70	p	97
q	71	q	98
r	72	r	99
s	73	s	A2
t	74	t	A3
u	75	u	A4
v	76	v	A5
w	77	w	A6
x	78	x	A7
y	79	y	A8
z	7A	z	A9
{	7B	{	43
|	7C	|	BB
}	7D	}	47
~	7E	~	DC
DEL	7F	DEL	07
	80	DS	20
	81	SOS	21
BPH	82	FS	22
NBH	83	WUS	23

ISO 8859-1 Data		CP278SWEDEN Translation	
Control/Graphic	Hex Value	Character	Hex Value
IND	84	BYP/INP	24
NEL	85	NL	15
SSA	86	RNL	06
ESA	87	POC	17
HTS	88	SA	28
HTJ	89	SFE	29
VTS	8A	SM/SW	2A
PLD	8B	CSP	2B
PLU	8C	MFA	2C
RI	8D	SPS	09
SS2	8E	RPT	0A
SS3	8F	CU1	1B
DCS	90		30
PU1	91		31
PU2	92	UBS	1A
STS	93	IR	33
CCH	94	PP	34
MW	95	TRN	35
SBA	96	NBS	36
EPA	97	GE	08
SOS	98	SBS	38
	99	IT	39
SCI	9A	RFF	3A
CSI	9B	CU3	3B
ST	9C	SEL	04
OSC	9D	RES/ENP	14
PM	9E		3E
APC	9F	EO	FF

ISO 8859-1 Data		CP278SWEDEN Translation	
Control/Graphic	Hex Value	Character	Hex Value
NBSP	A0	NBSP	41
¡	A1	¡	AA
¢	A2	¢	B0
£	A3	£	B1
¤	A4	¤	5A
¥	A5	¥	B2
¦	A6	¦	CC
§	A7	§	4A
¨	A8	¨	BD
©	A9	©	B4
ª	AA	ª	9A
«	AB	«	8A
¬	AC	¬	BA
-	AD	-	CA
®	AE	®	AF
¯	AF	¯	BC
°	B0	°	90
±	B1	±	8F
²	B2	²	EA
³	B3	³	FA
´	B4	´	BE
µ	B5	µ	A0
¶	B6	¶	B6
·	B7	·	B3
¸	B8	¸	9D
¹	B9	¹	DA
º	BA	º	9B
»	BB	»	8B

ISO 8859-1 Data		CP278SWEDEN Translation	
Control/Graphic	Hex Value	Character	Hex Value
¼	BC	¼	B7
½	BD	½	B8
¾	BE	¾	B9
¿	BF	¿	AB
À	C0	À	64
Á	C1	Á	65
Â	C2	Â	62
Ã	C3	Ã	66
Ä	C4	Ä	7B
Å	C5	Å	5B
Æ	C6	Æ	9E
Ç	C7	Ç	68
È	C8	È	74
É	C9	É	E0
Ê	CA	Ê	72
Ë	CB	Ë	73
Ì	CC	Ì	78
Í	CD	Í	75
Î	CE	Î	76
Ï	CF	Ï	77
Ð	D0	Ð	AC
Ñ	D1	Ñ	69
Ò	D2	Ò	ED
Ó	D3	Ó	EE
Ô	D4	Ô	EB
Õ	D5	Õ	EF
Ö	D6	Ö	7C
×	D7	×	BF

ISO 8859-1 Data		CP278SWEDEN Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ø	D8	Ø	80
Ù	D9	Ù	FD
Ú	DA	Ú	FE
Û	DB	Û	FB
Ü	DC	Ü	FC
Ý	DD	Ý	AD
Þ	DE	Þ	AE
ß	DF	ß	59
à	E0	à	44
á	E1	á	45
â	E2	â	42
ã	E3	ã	46
ä	E4	ä	C0
å	E5	å	D0
æ	E6	æ	9C
ç	E7	ç	48
è	E8	è	54
é	E9	é	79
ê	EA	ê	52
ë	EB	ë	53
ì	EC	ì	58
í	ED	í	55
î	EE	î	56
ï	EF	ï	57
ð	F0	ð	8C
ñ	F1	ñ	49
ò	F2	ò	CD
ó	F3	ó	CE

ISO 8859-1 Data		CP278SWEDEN Translation	
Control/Graphic	Hex Value	Character	Hex Value
ô	F4	ô	CB
õ	F5	õ	CF
ö	F6	ö	6A
÷	F7	÷	E1
ø	F8	ø	70
ù	F9	ù	DD
ú	FA	ú	DE
û	FB	û	DB
ü	FC	ü	A1
ý	FD	ý	8D
þ	FE	þ	8E
ÿ	FF	ÿ	DF

ISO 8859-1 8-bit ASCII to CP280ITALY

The following table shows ISO 8859-1 8-bit ASCII to Code Page 280 translation.

ISO 8859-1 Data		CP280ITALY Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
EOT	04	EOT	37
ENQ	05	ENQ	2D
ACK	06	ACK	2E
BEL	07	BEL	2F
BS	08	BS	16
HT	09	HT	05
LF	0A	LF	25
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO/LS1	0E	SO	0E
SI/LS0	0F	SI	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
DC4	14	DC4	3C
NAK	15	NAK	3D
SYN	16	SYN	32
ETB	17	ETB	26

ISO 8859-1 Data		CP280ITALY Translation	
Control/Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
SUB	1A	SUB	3F
ESC	1B	ESC	27
IFS	1C	IFS	1C
GS	1D	IGS	1D
RS	1E	IRS	1E
US	1F	IUS	1F
Space	20	Space	40
!	21	!	4F
"	22	"	7F
#	23	#	B1
$	24	$	5B
%	25	%	6C
&	26	&	50
'	27	'	7D
(	28	(	4D
)	29	)	5D
*	2A	*	5C
+	2B	+	4E
,	2C	,	6B
-	2D	-	60
.	2E	.	4B
/	2F	/	61

ISO 8859-1 Data		CP280ITALY Translation	
Control/Graphic	Hex Value	Character	Hex Value
0	30	0	F0
1	31	1	F1
2	32	2	F2
3	33	3	F3
4	34	4	F4
5	35	5	F5
6	36	6	F6
7	37	7	F7
8	38	8	F8
9	39	9	F9
:	3A	:	7A
;	3B	;	5E
<	3C	<	4C
=	3D	=	7E
>	3E	>	6E
?	3F	?	6F
@	40	@	B5
A	41	A	C1
B	42	B	C2
C	43	C	C3
D	44	D	C4
E	45	E	C5
F	46	F	C6
G	47	G	C7
H	48	H	C8
I	49	I	C9
J	4A	J	D1
K	4B	K	D2

ISO 8859-1 Data		CP280ITALY Translation	
Control/Graphic	Hex Value	Character	Hex Value
L	4C	L	D3
M	4D	M	D4
N	4E	N	D5
O	4F	O	D6
P	50	P	D7
Q	51	Q	D8
R	52	R	D9
S	53	S	E2
T	54	T	E3
U	55	U	E4
V	56	V	E5
W	57	W	E6
X	58	X	E7
Y	59	Y	E8
Z	5A	Z	E9
[	5B	[	90
\	5C	\	48
]	5D	]	51
^	5E	^	5F
_	5F	_	6D
`	60	`	DD
a	61	a	81
b	62	b	82
c	63	c	83
d	64	d	84
e	65	e	85
f	66	f	86
g	67	g	87

ISO 8859-1 Data		CP280ITALY Translation	
Control/Graphic	Hex Value	Character	Hex Value
h	68	h	88
i	69	i	89
j	6A	j	91
k	6B	k	92
l	6C	l	93
m	6D	m	94
n	6E	n	95
o	6F	o	96
p	70	p	97
q	71	q	98
r	72	r	99
s	73	s	A2
t	74	t	A3
u	75	u	A4
v	76	v	A5
w	77	w	A6
x	78	x	A7
y	79	y	A8
z	7A	z	A9
{	7B	{	44
|	7C	|	BB
}	7D	}	54
~	7E	~	58
DEL	7F	DEL	07
	80	DS	20
	81	SOS	21
BPH	82	FS	22
NBH	83	WUS	23

ISO 8859-1 Data		CP280ITALY Translation	
Control/Graphic	Hex Value	Character	Hex Value
IND	84	BYP/INP	24
NEL	85	NL	15
SSA	86	RNL	06
ESA	87	POC	17
HTS	88	SA	28
HTJ	89	SFE	29
VTS	8A	SM/SW	2A
PLD	8B	CSP	2B
PLU	8C	MFA	2C
RI	8D	SPS	09
SS2	8E	RPT	0A
SS3	8F	CU1	1B
DCS	90		30
PU1	91		31
PU2	92	UBS	1A
STS	93	IR	33
CCH	94	PP	34
MW	95	TRN	35
SBA	96	NBS	36
EPA	97	GE	08
SOS	98	SBS	38
	99	IT	39
SCI	9A	RFF	3A
CSI	9B	CU3	3B
ST	9C	SEL	04
OSC	9D	RES/ENP	14
PM	9E		3E
APC	9F	EO	FF

ISO 8859-1 Data		CP280ITALY Translation	
Control/Graphic	Hex Value	Character	Hex Value
NBSP	A0	NBSP	41
¡	A1	¡	AA
¢	A2	¢	B0
£	A3	£	7B
¤	A4	¤	9F
¥	A5	¥	B2
¦	A6	¦	CD
§	A7	§	7C
¨	A8	¨	BD
©	A9	©	B4
ª	AA	ª	9A
«	AB	«	8A
¬	AC	¬	BA
-	AD	-	CA
®	AE	®	AF
_	AF	_	BC
°	B0	°	4A
±	B1	±	8F
²	B2	²	EA
³	B3	³	FA
´	B4	´	BE
µ	B5	µ	A0
¶	B6	¶	B6
·	B7	·	B3
¸	B8	¸	9D
¹	B9	¹	DA
º	BA	º	9B
»	BB	»	8B

ISO 8859-1 Data		CP280ITALY Translation	
Control/Graphic	Hex Value	Character	Hex Value
¼	BC	¼	B7
½	BD	½	B8
¾	BE	¾	B9
¿	BF	¿	AB
À	C0	À	64
Á	C1	Á	65
Â	C2	Â	62
Ã	C3	Ã	66
Ä	C4	Ä	63
Å	C5	Å	67
Æ	C6	Æ	9E
Ç	C7	Ç	68
È	C8	È	74
É	C9	É	71
Ê	CA	Ê	72
Ë	CB	Ë	73
Ì	CC	Ì	78
Í	CD	Í	75
Î	CE	Î	76
Ï	CF	Ï	77
Ð	D0	Ð	AC
Ñ	D1	Ñ	69
Ò	D2	Ò	ED
Ó	D3	Ó	EE
Ô	D4	Ô	EB
Õ	D5	Õ	EF
Ö	D6	Ö	EC
×	D7	×	BF

ISO 8859-1 Data		CP280ITALY Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ø	D8	Ø	80
Ù	D9	Ù	FD
Ú	DA	Ú	FE
Û	DB	Û	FB
Ü	DC	Ü	FC
Ý	DD	Ý	AD
Þ	DE	Þ	AE
ß	DF	ß	59
à	E0	à	C0
á	E1	á	45
â	E2	â	42
ã	E3	ã	46
ä	E4	ä	43
å	E5	å	47
æ	E6	æ	9C
ç	E7	ç	E0
è	E8	è	D0
é	E9	é	5A
ê	EA	ê	52
ë	EB	ë	53
ì	EC	ì	A1
í	ED	í	55
î	EE	î	56
ï	EF	ï	57
ð	F0	ð	8C
ñ	F1	ñ	49
ò	F2	ò	6A
ó	F3	ó	CE

ISO 8859-1 Data		CP280ITALY Translation	
Control/Graphic	Hex Value	Character	Hex Value
ô	F4	ô	CB
õ	F5	õ	CF
ö	F6	ö	CC
÷	F7	÷	E1
ø	F8	ø	70
ù	F9	ù	79
ú	FA	ú	DE
û	FB	û	DB
ü	FC	ü	DC
ý	FD	ý	8D
þ	FE	þ	8E
ÿ	FF	ÿ	DF

ISO 8859-1 8-bit ASCII to CP284SPAIN

The following table shows ISO 8859-1 8-bit ASCII to Code Page 284 translation.

ISO 8859-1 Data		CP284SPAIN Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
EOT	04	EOT	37
ENQ	05	ENQ	2D
ACK	06	ACK	2E
BEL	07	BEL	2F
BS	08	BS	16
HT	09	HT	05
LF	0A	LF	25
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO/LS1	0E	SO	0E
SI/LS0	0F	SI	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
DC4	14	DC4	3C
NAK	15	NAK	3D
SYN	16	SYN	32
ETB	17	ETB	26

ISO 8859-1 Data		CP284SPAIN Translation	
Control/Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
SUB	1A	SUB	3F
ESC	1B	ESC	27
IFS	1C	IFS	1C
GS	1D	IGS	1D
RS	1E	IRS	1E
US	1F	IUS	1F
Space	20	Space	40
!	21	!	BB
"	22	"	7F
#	23	#	69
$	24	$	5B
%	25	%	6C
&	26	&	50
'	27	'	7D
(	28	(	4D
)	29	)	5D
*	2A	*	5C
+	2B	+	4E
,	2C	,	6B
-	2D	-	60
.	2E	.	4B
/	2F	/	61

ISO 8859-1 Data		CP284SPAIN Translation	
Control/Graphic	Hex Value	Character	Hex Value
0	30	0	F0
1	31	1	F1
2	32	2	F2
3	33	3	F3
4	34	4	F4
5	35	5	F5
6	36	6	F6
7	37	7	F7
8	38	8	F8
9	39	9	F9
:	3A	:	7A
;	3B	;	5E
<	3C	<	4C
=	3D	=	7E
>	3E	>	6E
?	3F	?	6F
@	40	@	7C
A	41	A	C1
B	42	B	C2
C	43	C	C3
D	44	D	C4
E	45	E	C5
F	46	F	C6
G	47	G	C7
H	48	H	C8
I	49	I	C9
J	4A	J	D1
K	4B	K	D2

ISO 8859-1 Data		CP284SPAIN Translation	
Control/Graphic	Hex Value	Character	Hex Value
L	4C	L	D3
M	4D	M	D4
N	4E	N	D5
O	4F	O	D6
P	50	P	D7
Q	51	Q	D8
R	52	R	D9
S	53	S	E2
T	54	T	E3
U	55	U	E4
V	56	V	E5
W	57	W	E6
X	58	X	E7
Y	59	Y	E8
Z	5A	Z	E9
[	5B	[	4A
\	5C	\	E0
]	5D	]	5A
^	5E	^	BA
_	5F	_	6D
`	60	`	79
a	61	a	81
b	62	b	82
c	63	c	83
d	64	d	84
e	65	e	85
f	66	f	86
g	67	g	87

ISO 8859-1 Data		CP284SPAIN Translation	
Control/Graphic	Hex Value	Character	Hex Value
h	68	h	88
i	69	i	89
j	6A	j	91
k	6B	k	92
l	6C	l	93
m	6D	m	94
n	6E	n	95
o	6F	o	96
p	70	p	97
q	71	q	98
r	72	r	99
s	73	s	A2
t	74	t	A3
u	75	u	A4
v	76	v	A5
w	77	w	A6
x	78	x	A7
y	79	y	A8
z	7A	z	A9
{	7B	{	C0
|	7C	|	4F
}	7D	}	D0
~	7E	~	BD
DEL	7F	DEL	07
	80	DS	20
	81	SOS	21
BPH	82	FS	22
NBH	83	WUS	23

ISO 8859-1 Data		CP284SPAIN Translation	
Control/Graphic	Hex Value	Character	Hex Value
IND	84	BYP/INP	24
NEL	85	NL	15
SSA	86	RNL	06
ESA	87	POC	17
HTS	88	SA	28
HTJ	89	SFE	29
VTS	8A	SM/SW	2A
PLD	8B	CSP	2B
PLU	8C	MFA	2C
RI	8D	SPS	09
SS2	8E	RPT	0A
SS3	8F	CU1	1B
DCS	90		30
PU1	91		31
PU2	92	UBS	1A
STS	93	IR	33
CCH	94	PP	34
MW	95	TRN	35
SBA	96	NBS	36
EPA	97	GE	08
SOS	98	SBS	38
	99	IT	39
SCI	9A	RFF	3A
CSI	9B	CU3	3B
ST	9C	SEL	04
OSC	9D	RES/ENP	14
PM	9E		3E
APC	9F	EO	FF

ISO 8859-1 Data		CP284SPAIN Translation	
Control/Graphic	Hex Value	Character	Hex Value
NBSP	A0	NBSP	41
¡	A1	¡	AA
¢	A2	¢	B0
£	A3	£	B1
¤	A4	¤	9F
¥	A5	¥	B2
¦	A6	¦	49
§	A7	§	B5
¨	A8	¨	A1
©	A9	©	B4
ª	AA	ª	9A
«	AB	«	8A
¬	AC	¬	5F
-	AD	-	CA
®	AE	®	AF
_	AF	_	BC
°	B0	°	90
±	B1	±	8F
²	B2	²	EA
³	B3	³	FA
´	B4	´	BE
µ	B5	µ	A0
¶	B6	¶	B6
·	B7	·	B3
¸	B8	¸	9D
¹	B9	¹	DA
º	BA	º	9B
»	BB	»	8B

ISO 8859-1 Data		CP284SPAIN Translation	
Control/Graphic	Hex Value	Character	Hex Value
¼	BC	¼	B7
½	BD	½	B8
¾	BE	¾	B9
¿	BF	¿	AB
À	C0	À	64
Á	C1	Á	65
Â	C2	Â	62
Ã	C3	Ã	66
Ä	C4	Ä	63
Å	C5	Å	67
Æ	C6	Æ	9E
Ç	C7	Ç	68
È	C8	È	74
É	C9	É	71
Ê	CA	Ê	72
Ë	CB	Ë	73
Ì	CC	Ì	78
Í	CD	Í	75
Î	CE	Î	76
Ï	CF	Ï	77
Ð	D0	Ð	AC
Ñ	D1	Ñ	7B
Ò	D2	Ò	ED
Ó	D3	Ó	EE
Ô	D4	Ô	EB
Õ	D5	Õ	EF
Ö	D6	Ö	EC
×	D7	×	BF

ISO 8859-1 Data		CP284SPAIN Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ø	D8	Ø	80
Ù	D9	Ù	FD
Ú	DA	Ú	FE
Û	DB	Û	FB
Ü	DC	Ü	FC
Ý	DD	Ý	AD
Þ	DE	Þ	AE
ß	DF	ß	59
à	E0	à	44
á	E1	á	45
â	E2	â	42
ã	E3	ã	46
ä	E4	ä	43
å	E5	å	47
æ	E6	æ	9C
ç	E7	ç	48
è	E8	è	54
é	E9	é	51
ê	EA	ê	52
ë	EB	ë	53
ì	EC	ì	58
í	ED	í	55
î	EE	î	56
ï	EF	ï	57
ð	F0	ð	8C
ñ	F1	ñ	6A
ò	F2	ò	CD
ó	F3	ó	CE

ISO 8859-1 Data		CP284SPAIN Translation	
Control/Graphic	Hex Value	Character	Hex Value
ô	F4	ô	CB
õ	F5	õ	CF
ö	F6	ö	CC
÷	F7	÷	E1
ø	F8	ø	70
ù	F9	ù	DD
ú	FA	ú	DE
û	FB	û	DB
ü	FC	ü	DC
ý	FD	ý	8D
þ	FE	þ	8E
ÿ	FF	ÿ	DF

ISO 8859-1 8-bit ASCII to CP285UK

The following table shows ISO 8859-1 8-bit ASCII to Code Page 285 translation.

ISO 8859-1 Data		CP285UK Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
EOT	04	EOT	37
ENQ	05	ENQ	2D
ACK	06	ACK	2E
BEL	07	BEL	2F
BS	08	BS	16
HT	09	HT	05
LF	0A	LF	25
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO/LS1	0E	SO	0E
SI/LS0	0F	SI	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
DC4	14	DC4	3C
NAK	15	NAK	3D
SYN	16	SYN	32
ETB	17	ETB	26
CAN	18	CAN	18

ISO 8859-1 Data		CP285UK Translation	
Control/Graphic	Hex Value	Character	Hex Value
EM	19	EM	19
SUB	1A	SUB	3F
ESC	1B	ESC	27
IFS	1C	IFS	1C
GS	1D	IGS	1D
RS	1E	IRS	1E
US	1F	IUS	1F
Space	20	Space	40
!	21	!	5A
"	22	"	7F
#	23	#	7B
$	24	$	4A
%	25	%	6C
&	26	&	50
'	27	'	7D
(	28	(	4D
)	29	)	5D
*	2A	*	5C
+	2B	+	4E
,	2C	,	6B
-	2D	-	60
.	2E	.	4B
/	2F	/	61
0	30	0	F0
1	31	1	F1

ISO 8859-1 Data		CP285UK Translation	
Control/Graphic	Hex Value	Character	Hex Value
2	32	2	F2
3	33	3	F3
4	34	4	F4
5	35	5	F5
6	36	6	F6
7	37	7	F7
8	38	8	F8
9	39	9	F9
:	3A	:	7A
;	3B	;	5E
<	3C	<	4C
=	3D	=	7E
>	3E	>	6E
?	3F	?	6F
@	40	@	7C
A	41	A	C1
B	42	B	C2
C	43	C	C3
D	44	D	C4
E	45	E	C5
F	46	F	C6
G	47	G	C7
H	48	H	C8
I	49	I	C9
J	4A	J	D1
K	4B	K	D2
L	4C	L	D3
M	4D	M	D4

ISO 8859-1 Data		CP285UK Translation	
Control/Graphic	Hex Value	Character	Hex Value
N	4E	N	D5
O	4F	O	D6
P	50	P	D7
Q	51	Q	D8
R	52	R	D9
S	53	S	E2
T	54	T	E3
U	55	U	E4
V	56	V	E5
W	57	W	E6
X	58	X	E7
Y	59	Y	E8
Z	5A	Z	E9
[	5B	[	B1
\	5C	\	E0
]	5D	]	BB
^	5E	^	BA
_	5F	_	6D
`	60	`	79
a	61	a	81
b	62	b	82
c	63	c	83
d	64	d	84
e	65	e	85
f	66	f	86
g	67	g	87
h	68	h	88
i	69	i	89

ISO 8859-1 Data		CP285UK Translation	
Control/Graphic	Hex Value	Character	Hex Value
j	6A	j	91
k	6B	k	92
l	6C	l	93
m	6D	m	94
n	6E	n	95
o	6F	o	96
p	70	p	97
q	71	q	98
r	72	r	99
s	73	s	A2
t	74	t	A3
u	75	u	A4
v	76	v	A5
w	77	w	A6
x	78	x	A7
y	79	y	A8
z	7A	z	A9
{	7B	{	C0
|	7C	|	4F
}	7D	}	D0
~	7E	~	BC
DEL	7F	DEL	07
	80	DS	20
	81	SOS	21
BPH	82	FS	22
NBH	83	WUS	23
IND	84	BYP/INP	24
NEL	85	NL	15

ISO 8859-1 Data		CP285UK Translation	
Control/Graphic	Hex Value	Character	Hex Value
SSA	86	RNL	06
ESA	87	POC	17
HTS	88	SA	28
HTJ	89	SFE	29
VTS	8A	SM/SW	2A
PLD	8B	CSP	2B
PLU	8C	MFA	2C
RI	8D	SPS	09
SS2	8E	RPT	0A
SS3	8F	CU1	1B
DCS	90		30
PU1	91		31
PU2	92	UBS	1A
STS	93	IR	33
CCH	94	PP	34
MW	95	TRN	35
SBA	96	NBS	36
EPA	97	GE	08
SOS	98	SBS	38
	99	IT	39
SCI	9A	RFF	3A
CSI	9B	CU3	3B
ST	9C	SEL	04
OSC	9D	RES/ENP	14
PM	9E		3E
APC	9F	EO	FF
NBSP	A0	NBSP	41
¡	A1	¡	AA

ISO 8859-1 Data		CP285UK Translation	
Control/Graphic	Hex Value	Character	Hex Value
¢	A2	¢	B0
£	A3	£	5B
¤	A4	¤	9F
¥	A5	¥	B2
¦	A6	¦	6A
§	A7	§	B5
¨	A8	¨	BD
©	A9	©	B4
ª	AA	ª	9A
«	AB	«	8A
¬	AC	¬	5F
-	AD	-	CA
®	AE	®	AF
¯	AF	¯	A1
°	B0	°	90
±	B1	±	8F
²	B2	²	EA
³	B3	³	FA
´	B4	´	BE
µ	B5	µ	A0
¶	B6	¶	B6
·	B7	·	B3
¸	B8	¸	9D
¹	B9	¹	DA
º	BA	º	9B
»	BB	»	8B
¼	BC	¼	B7
½	BD	½	B8

ISO 8859-1 Data		CP285UK Translation	
Control/Graphic	Hex Value	Character	Hex Value
¾	BE	¾	B9
¿	BF	¿	AB
À	C0	À	64
Á	C1	Á	65
Â	C2	Â	62
Ã	C3	Ã	66
Ä	C4	Ä	63
Å	C5	Å	67
Æ	C6	Æ	9E
Ç	C7	Ç	68
È	C8	È	74
É	C9	É	71
Ê	CA	Ê	72
Ë	CB	Ë	73
Ì	CC	Ì	78
Í	CD	Í	75
Î	CE	Î	76
Ï	CF	Ï	77
Ð	D0	Ð	AC
Ñ	D1	Ñ	69
Ò	D2	Ò	ED
Ó	D3	Ó	EE
Ô	D4	Ô	EB
Õ	D5	Õ	EF
Ö	D6	Ö	EC
×	D7	×	BF
Ø	D8	Ø	80
Ù	D9	Ù	FD

ISO 8859-1 Data		CP285UK Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ú	DA	Ú	FE
Û	DB	Û	FB
Ü	DC	Ü	FC
Ý	DD	Ý	AD
Þ	DE	Þ	AE
ß	DF	ß	59
à	E0	à	44
á	E1	á	45
â	E2	â	42
ã	E3	ã	46
ä	E4	ä	43
å	E5	å	47
æ	E6	æ	9C
ç	E7	ç	48
è	E8	è	54
é	E9	é	51
ê	EA	ê	52
ë	EB	ë	53
ì	EC	ì	58
í	ED	í	55
î	EE	î	56
ï	EF	ï	57
ð	F0	ð	8C
ñ	F1	ñ	49
ò	F2	ò	CD
ó	F3	ó	CE
ô	F4	ô	CB
õ	F5	õ	CF

ISO 8859-1 Data		CP285UK Translation	
Control/Graphic	Hex Value	Character	Hex Value
ö	F6	ö	CC
÷	F7	÷	E1
ø	F8	ø	70
ù	F9	ù	DD
ú	FA	ú	DE
û	FB	û	DB
ü	FC	ü	DC
ý	FD	ý	8D
þ	FE	þ	8E
ÿ	FF	ÿ	DF

ISO 8859-1 8-bit ASCII to CP871ICELAND

The following table shows ISO 8859-1 8-bit ASCII to Code Page 871 translation.

ISO 8859-1 Data		CP871ICELAND Translation	
Control/Graphic	Hex Value	Character	Hex Value
NUL	00	NUL	00
SOH	01	SOH	01
STX	02	STX	02
ETX	03	ETX	03
EOT	04	EOT	37
ENQ	05	ENQ	2D
ACK	06	ACK	2E
BEL	07	BEL	2F
BS	08	BS	16
HT	09	HT	05
LF	0A	LF	25
VT	0B	VT	0B
FF	0C	FF	0C
CR	0D	CR	0D
SO/LS1	0E	SO	0E
SI/LS0	0F	SI	0F
DLE	10	DLE	10
DC1	11	DC1	11
DC2	12	DC2	12
DC3	13	DC3	13
DC4	14	DC4	3C
NAK	15	NAK	3D
SYN	16	SYN	32
ETB	17	ETB	26

ISO 8859-1 Data		CP871ICELAND Translation	
Control/Graphic	Hex Value	Character	Hex Value
CAN	18	CAN	18
EM	19	EM	19
SUB	1A	SUB	3F
ESC	1B	ESC	27
IFS	1C	IFS	1C
GS	1D	IGS	1D
RS	1E	IRS	1E
US	1F	IUS	1F
Space	20	Space	40
!	21	!	4F
"	22	"	7F
#	23	#	7B
$	24	$	5B
%	25	%	6C
&	26	&	50
'	27	'	7D
(	28	(	4D
)	29	)	5D
*	2A	*	5C
+	2B	+	4E
,	2C	,	6B
-	2D	-	60
.	2E	.	4B
/	2F	/	61

ISO 8859-1 Data		CP871ICELAND Translation	
Control/Graphic	Hex Value	Character	Hex Value
0	30	0	F0
1	31	1	F1
2	32	2	F2
3	33	3	F3
4	34	4	F4
5	35	5	F5
6	36	6	F6
7	37	7	F7
8	38	8	F8
9	39	9	F9
:	3A	:	7A
;	3B	;	5E
<	3C	<	4C
=	3D	=	7E
>	3E	>	6E
?	3F	?	6F
@	40	@	AC
A	41	A	C1
B	42	B	C2
C	43	C	C3
D	44	D	C4
E	45	E	C5
F	46	F	C6
G	47	G	C7
H	48	H	C8
I	49	I	C9
J	4A	J	D1

ISO 8859-1 Data		CP871ICELAND Translation	
Control/Graphic	Hex Value	Character	Hex Value
K	4B	K	D2
L	4C	L	D3
M	4D	M	D4
N	4E	N	D5
O	4F	O	D6
P	50	P	D7
Q	51	Q	D8
R	52	R	D9
S	53	S	E2
T	54	T	E3
U	55	U	E4
V	56	V	E5
W	57	W	E6
X	58	X	E7
Y	59	Y	E8
Z	5A	Z	E9
[	5B	[	AE
\	5C	\	BE
]	5D	]	9E
^	5E	^	EC
_	5F	_	6D
`	60	`	8C
a	61	a	81
b	62	b	82
c	63	c	83
d	64	d	84
e	65	e	85

ISO 8859-1 Data		CP871ICELAND Translation	
Control/Graphic	Hex Value	Character	Hex Value
f	66	f	86
g	67	g	87
h	68	h	88
i	69	i	89
j	6A	j	91
k	6B	k	92
l	6C	l	93
m	6D	m	94
n	6E	n	95
o	6F	o	96
p	70	p	97
q	71	q	98
r	72	r	99
s	73	s	A2
t	74	t	A3
u	75	u	A4
v	76	v	A5
w	77	w	A6
x	78	x	A7
y	79	y	A8
z	7A	z	A9
{	7B	{	8E
|	7C	|	BB
}	7D	}	9C
~	7E	~	CC
DEL	7F	DEL	07
	80	DS	20
	81	SOS	21
BPH	82	FS	22
NBH	83	WUS	23
IND	84	BYP/INP	24
NEL	85	NL	15
SSA	86	RNL	06
ESA	87	POC	17
HTS	88	SA	28
HTJ	89	SFE	29
VTS	8A	SM/SW	2A
PLD	8B	CSP	2B
PLU	8C	MFA	2C
RI	8D	SPS	09
SS2	8E	RPT	0A
SS3	8F	CU1	1B
DCS	90		30
PU1	91		31
PU2	92	UBS	1A
STS	93	IR	33
CCH	94	PP	34
MW	95	TRN	35
SBA	96	NBS	36
EPA	97	GE	08
SOS	98	SBS	38
	99	IT	39
SCI	9A	RFF	3A
CSI	9B	CU3	3B

ISO 8859-1 Data		CP871ICELAND Translation	
Control/Graphic	Hex Value	Character	Hex Value
ST	9C	SEL	04
OSC	9D	RES/ENP	14
PM	9E		3E
APC	9F	EO	FF
NBSP	A0	NBSP	41
¡	A1	¡	AA
¢	A2	¢	B0
£	A3	£	B1
¤	A4	¤	9F
¥	A5	¥	B2
¦	A6	¦	6A
§	A7	§	B5
¨	A8	¨	BD
©	A9	©	B4
ª	AA	ª	9A
«	AB	«	8A
¬	AC	¬	BA
-	AD	-	CA
®	AE	®	AF
¯	AF	¯	BC
°	B0	°	90
±	B1	±	8F
²	B2	²	EA
³	B3	³	FA
´	B4	´	E0
µ	B5	µ	A0
¶	B6	¶	B6

ISO 8859-1 Data		CP871ICELAND Translation	
Control/Graphic	Hex Value	Character	Hex Value
·	B7	·	B3
¸	B8	¸	9D
¹	B9	¹	DA
º	BA	º	9B
»	BB	»	8B
¼	BC	¼	B7
½	BD	½	B8
¾	BE	¾	B9
¿	BF	¿	AB
À	C0	À	64
Á	C1	Á	65
Â	C2	Â	62
Ã	C3	Ã	66
Ä	C4	Ä	63
Å	C5	Å	67
Æ	C6	Æ	5A
Ç	C7	Ç	68
È	C8	È	74
É	C9	É	71
Ê	CA	Ê	72
Ë	CB	Ë	73
Ì	CC	Ì	78
Í	CD	Í	75
Î	CE	Î	76
Ï	CF	Ï	77
Ð	D0	Ð	7C
Ñ	D1	Ñ	69

ISO 8859-1 Data		CP871ICELAND Translation	
Control/Graphic	Hex Value	Character	Hex Value
Ò	D2	Ò	ED
Ó	D3	Ó	EE
Ô	D4	Ô	EB
Õ	D5	Õ	EF
Ö	D6	Ö	5F
×	D7	×	BF
Ø	D8	Ø	80
Ù	D9	Ù	FD
Ú	DA	Ú	FE
Û	DB	Û	FB
Ü	DC	Ü	FC
Ý	DD	Ý	AD
Þ	DE	Þ	4A
ß	DF	ß	59
à	E0	à	44
á	E1	á	45
â	E2	â	42
ã	E3	ã	46
ä	E4	ä	43
å	E5	å	47
æ	E6	æ	D0
ç	E7	ç	48
è	E8	è	54
é	E9	é	51
ê	EA	ê	52
ë	EB	ë	53
ì	EC	ì	58

ISO 8859-1 Data		CP871ICELAND Translation	
Control/Graphic	Hex Value	Character	Hex Value
í	ED	í	55
î	EE	î	56
ï	EF	ï	57
ð	F0	ð	79
ñ	F1	ñ	49
ò	F2	ò	CD
ó	F3	ó	CE
ô	F4	ô	CB
õ	F5	õ	CF
ö	F6	ö	A1
÷	F7	÷	E1
ø	F8	ø	70
ù	F9	ù	DD
ú	FA	ú	DE
û	FB	û	DB
ü	FC	ü	DC
ý	FD	ý	8D
þ	FE	þ	C0
ÿ	FF	ÿ	DF

Appendix D. Handling of COBOL Binary Data

Your UNIX COBOL compiler may differ from your mainframe COBOL compiler in the size, storage sequence and alignment of binary data fields.

If your mainframe compiler is VS COBOL II, your UNIX compiler is Micro Focus COBOL, you use the IBMCOMP Micro Focus compiler directive, and your UNIX system uses big-endian representation for binary data then there will generally be no difference in this area.

If you use something other than the above, then you may need to convert your data differently by specifying additional arguments to FilePort and/or amending your COBOL data definitions before compiling.

Field Size

Fixed Point Data

The VS COBOL II compiler assigns 2, 4, or 8 bytes of storage to binary data items (USAGE IS BINARY and its synonyms), depending on whether there are 1-4, 5-9, or 10-18 digits in the PICTURE clause.

If your UNIX compiler does not assign the same storage (for example, it may allocate a smaller field that is still large enough to contain the specified precision), you will need to modify your copybook to make the definitions match. Your COBOL documentation provides details.

Floating Point Data

The VS COBOL II compiler assigns either 4 or 8 bytes of storage to floating point data items (USAGE IS COMP-1, COMP-2). FilePort assumes that you use double precision (8 byte) floating point fields on the UNIX system since this is the largest format supported, and single precision floating point may lose precision in the translation. Your input and output files may therefore be different sizes for this reason.

Storage Sequence (Big-endian or Little-endian)

Mainframe binary data is stored in big-endian format, i.e. the data field address points to the highest order byte in the field.

UNIX systems based on processors such as Hewlett-Packard PA-RISC (HP-UX), Sun SPARC (SunOS, Solaris), IBM POWER (AIX), and MIPS (DC/OSx, SINIX, IRIX) use big-endian representation. On these systems no special actions need to be taken because binary output data will automatically be produced in big-endian format.

UNIX systems based on processors such as Intel (DYNIX/ptx, SCO, SVR4 Intel, UnixWare), and Alpha (Digital UNIX, OSF/1) use little-endian representation, i.e. the data field address points to the lowest order byte in the field. On these systems binary output data will automatically be produced in little-endian format. To correctly access this data through Micro Focus COBOL you must change your BINARY, COMP and COMP-4 usage clauses to specify COMP-5.

Field Alignment and Slack Bytes

Slack bytes within records may be generated by the VS COBOL II compiler when the SYNCHRONIZED clause is specified for a BINARY, COMP, COMP-1, COMP-2, COMP-4, POINTER or INDEX data item. In addition, slack bytes may be generated when a group item containing a SYNCHRONIZED binary data item is specified with an OCCURS clause. The insertion of slack bytes is done to align a SYNCHRONIZED item on the appropriate natural boundary in memory for its data type. For example, a SYNCHRONIZED data item with level number > 1 and between 5 and 18 digits is aligned such that its location is a multiple of 4 bytes from the beginning of the record.

Slack bytes are not necessarily generated by the COBOL compiler you use on your UNIX system. If there are any slack bytes in your record layout, you need to determine if they have to be removed for processing on UNIX.

When your UNIX COBOL compiler has an option to either generate slack bytes or not, you have the choice of leaving the slack bytes in the records, or removing them. When your compiler does not have this option, you have to remove the slack bytes.

To request the removal of all slack bytes when converting from mainframe to UNIX, specify the **noslack** argument on the **-outfile** option. To indicate that slack bytes have been removed when converting from UNIX to mainframe, specify the **noslack** argument on the **-infile** option.

Appendix E. Identifiers and Field Names

Specifying Field Names

In many FilePort options you need to refer by name to fields which are defined elsewhere. Such fields may be defined in a **-recordlayout** or in a COBOL copybook defined through the **-datadictionary** option. These field names must conform to the rules for identifiers as follows:

Identifiers

An identifier in FilePort may contain letters, digits, underscores and hyphens. A hyphen can neither be the first nor the last character. An identifier cannot consist entirely of digits. Also, an identifier cannot be longer than 32 characters. Uppercase and lowercase characters are considered equivalent. For example, "CUSTOMER", "customer" and "Customer", are treated as the same identifier.

Examples of Valid Identifiers

```
customer
Company
CONVERSION_IND
_temp
s1_help
daytime
X
zp
A-FIELD
```

Examples of Invalid Identifiers

```
this_is_a_much_too_long_identifier
```

This is an invalid identifier because it contains more than 32 characters.

```
bad identifier
```

An identifier must not contain embedded spaces.

Qualified Field Names

In the majority of cases, a field can be referred to by a simple, unqualified name such as customer-name, date-received, total-amount, item-code, state, etc. In two circumstances the unqualified name may not be sufficient to identify a field unambiguously. The first is when the field is a subdivision of some higher level entity, i.e. it is a subfield. This is the case when a COBOL elementary item is subordinate to a group item. The second circumstance when an unqualified name is not sufficient is when the field is defined as a repeated field, i.e. it is an array. This is the case for the OCCURS clause in a COBOL record layout.

Sub-fields

There is effectively no limit to the level of nesting of fields within other fields. The highest level field is referred to by its unqualified name. Fields at all other levels may be qualified. Take as an example the record layout shown in Figure 1 on the next page.

All fields in this example are either composite fields or elementary fields which are components of a composite field. Several fields are subordinate to more than one composite field, such as the manager's first name which is a component of mng-name, which in turn is a component of manager. To refer to a field name in an option, you must provide a fully qualified name, or sufficient qualification that the name is not ambiguous. A fully qualified name consists of the name of the field and the names of composite fields of which it is a component, ordered from the highest level composite field to the field itself, left to right. The names are separated by a period. For example, the fully qualified name for the manager's first name is employee.manager.mng-name.first-name. You could also refer to this field as mng-name.first-name, manager.first-name and first-name. However, first-name is not sufficiently qualified to distinguish the manager's first name from the employee's first name so this form of the name is not acceptable.

```
01 EMPLOYEE.
      05 EMP-NAME.
            10 FIRST-NAME           PIC X(20).
            10 MIDDLE-INIT          PIC X.
            10 LAST-NAME            PIC X(20).
      05 HOME-ADDRESS.
            10 STREET               PIC X(30).
            10 CITY                 PIC X(30).
            10 STATE                PIC XX.
            10 ZIP                  PIC X(9).
      05 TELEPHONE.
            10 AREA-CODE            PIC 999.
            10 TEL-NUM              PIC 9(7).
      05 MAIL-ADDRESS.
            10 STREET               PIC X(30).
            10 CITY                 PIC X(30).
            10 STATE                PIC XX.
            10 ZIP                  PIC X(9).
      05 SS-NUM                     PIC 9(9).
      05 MANAGER.
            10 MNG-NAME.
                  15 FIRST-NAME     PIC X(20).
                  15 MIDDLE-INIT    PIC X.
                  15 LAST-NAME      PIC X(20).
            10 TELEPHONE.
                  15 AREA-CODE      PIC 999.
                  15 TEL-NUM        PIC 9(7).
            10 ASSIGNED             PIC 9(6).
```

Figure 1. Sample COBOL Record Layout

The minimum qualification of names for each field in the above record layout follow:

emp-name.first-name
emp-name.middle-init
emp-name.last-name
home-address.street
home-address.city
home-address.state
home-address.zip
telephone.area-code
telephone.tel-num
mail-address.street
mail-address.city
mail-address.state

mail-address.zip
ss-num
mng-name.first-name
mng-name.middle-init
mng-name.last-name
manager.area-code
manager.tel-num
assigned

Arrays

All fields in an array have the same data type, the same length, and the same name, which is referred to here as the array name. Both elementary fields and composite fields can form arrays. To reference a particular field in an array, you specify the name of the array followed by the number of the individual field, surrounded by brackets. The number portion of the field definition is called the subscript. The fields in the array are numbered sequentially from 1. Hence, the fifth field in the daily-total array is referenced by name as daily-total[5]. If you refer to a field with a subscript larger than the number of fields in the array, FilePort treats this as an undefined field.

Since composite fields can be formed into arrays, it follows that arrays can occur inside arrays. The 'multi-dimensioned' array complicates referencing individual fields. For a two dimensional array, you need to specify two subscripts to reference an individual field in the innermost array, e.g. trans-count[6, 31]. Referencing an individual field in a three dimensional array requires three subscripts, etc. The order for the subscripts is in descending order of composite level, from the outermost composite field.

Consider the record layout in the following example.

```
01 WEATHER-RECORDS.
      05 YEAR-OF-RECORD             PIC 9(4).
      05 LOCATION                   OCCURS 5 TIMES.
            10 LOCATION-NAME        PIC X(20).
            10 MONTHLY-DATA         OCCURS 12 TIMES.
                  15 TOTAL-RAIN     PIC 9(4) COMP.
                  15 DAILY-DATA     OCCURS 31 TIMES.
                        20 MAX-TEMP PIC 999.
                        20 MIN-TEMP PIC 999.
                        20 RAINFALL PIC 9(4) COMP.
```

In this example, location is a one dimensional array with 5 elements, monthly-data is a two dimensional array with 5 * 12 elements and daily-data is a three dimensional array with 5 * 12 * 31 elements. To reference the rainfall for the 21st day of the 3rd month at the 2nd location, you could specify rainfall[2, 3, 21] as the field name. The fully qualified field name is weather-records.location.monthly-data.daily-data.rainfall[2, 3, 21]. The monthly rainfall for the 12th month at the 1st location is referenced by total-rain[1, 12] or location.monthly-data.total-rain[1, 12]. The name of the 5th location is referenced by location-name[5] or location.location-name[5].

Appendix F. The Syncsort Message Utility

The Syncsort Message Utility, syncmsg, provides detailed information on messages that may be displayed when running Syncsort products.

The format of the **syncmsg** command is as follows:

syncmsg [*mnemonic*]

where

mnemonic — exception mnemonic appearing in parentheses preceding the text of any message.

Notes

When you enter a mnemonic with the **syncmsg** command, the message associated with the mnemonic is displayed along with a detailed explanation. In some cases, you are also given an action to correct the condition that caused the problem.

To exit from syncmsg, press <enter> without keying any characters.

Example

```
syncmsg floatinexp

FilePort : (FLOATINEXP) comparison with a floating point
field is not supported

EXPLANATION:
A condition which you have defined through a -condition
option requires a comparison involving a floating point
field. FilePort currently  does not support comparisons
involving IBM excess-64 floating point values.

ACTION:
If you are unable to express the condition without involving
the floating point field value, the conditional conversion
is currently not possible.
```

Appendix G. Technical Support

If you encounter difficulties in installing or running FilePort, you can reach a Syncsort engineer by telephone 24 hours a day, seven days a week.

Address	Syncsort Inc. Attention: UNIX Technical Support 50 Tice Boulevard Woodcliff Lake, NJ 07677 U.S.A.
Telephone	201-930-8270
Fax	201-930-8281
E-mail	unixtech@syncsort.com
Web site	http://www.syncsort.com

Index