DB/C Utilities 101 Guide and Reference

January 2023

Table of Contents

Runtime Properties

The DB/C DX runtime properties file contains configuration and runtime information for execution of DB/C Utilities as well as for DB/C DX programs. The properties file is a native text file that contains key word value pairs in each line of the file. Properties files use the backslash character (\) as an escape character. Thus when you need to include one backslash in a value, specify two consecutive backslashes.

This search for the runtime properties file is done in the following order:

1. **-cfg=***filename* option specified on the command line.
2. **DBC_CFG=***filename* defined in the operating system environment.
3. the file named **dbcdx.cfg** located in the current directory.

These elements are used in the following descriptions:

string is a string of characters and digits. It may contain blanks.

number is a string of one or more digits.

directory [**;** *directory* …] is one or more directories separated by a semicolon.

translate-spec is either *nnn***:***nnn* or *nnn***-***nnn***:***nnn* where *nnn* is the decimal value of a character. The value before the colon is the translate from character or the translate from character range, and the character after the colon is the translate to character or the first character in the translate to range.

The following keyword-value pairs of properties apply to the DB/C DX Utilities. Unless otherwise noted, each keyword may only be specified once in the properties file.

dbcdx.memalloc = *nnn*
This property specifies the number of kilobytes that will be allocated for utility. The default is 2048. This value should be increased if an out of memory error occurs.

dbcdx.file.casemap = *translate-spec* [**;** *translate-spec* …]
This property specifies the file case translation table used to modify the default behavior for case insensitive aim reads. This property may be needed to support international character set as DB/C DX by default only converts **a-z** to **A-Z** for aim record comparisons. This would be equivalent to having a translate value of 97-122 : 65.

dbcdx.file.collate = *translate-spec* [**;** *translate-spec* …]
This property specifies the collating sequence table used to modify the default behavior of the sort utility and index keys. This property may be needed to support international character sets as DB/C DX by default sorts using the ASCII value of the character. This would be equivalent to having a translate value of 0-255 : 0.

dbcdx.file.compat = dos
dbcdx.file.compat = rms
dbcdx.file.compat = rmsx
This property specifies the Datapoint DATABUS file statement compatibility. This property causes the file name specified on the command line of utilities to be altered before the operation occurs. If **dos** is specified, the first slash (/) in the file name is translated to a period. If **rms** is specified, the first slash (/) in the file name is translated to a period. If the extension is **TEXT**, it is translated to **TXT**. If the extension is **ISAM**, it is translated to **ISI**. In addition, if the name portion of the file name (left of the slash) is longer than eight characters, it is truncated to eight characters. If **rmsx** is specified, the same actions occur as for **rms**, except name portion truncation does not occur.

dbcdx.file.editcfg = *directory*
The source code editor utility (edit) stores configuration information in a file named **edit.cfg** in the current directory. If this runtime property is specified, the directory location of this file is designated by directory.

dbcdx.file.extcase = upper
This property specifies how default extensions are appended to file names that do not contain

an extension. If this property is specified, an uppercase extension is appended to the file name. If this property is not specified, a lowercase extension is appended to the file name.

dbcdx.file.ichrs = on

This property allows DB/C-type files to contain characters in the range 128 through 248. Digit compression is disabled, but space compression still works. If this property is not specified, then values 128 through 248 are considered to be digit compression characters.

dbcdx.file.namecase = upper
dbcdx.file.namecase = lower

If the value is a **upper**, then all file names are translated to upper case. If the value is **lower**, then all file names are translated to lower case.

dbcdx.file.open = *directory* [**;** *directory* …]

This property specifies the name of the local directories to search when opening a file. If the value is a directory or a list of directories an attempt is made to locate that file in the directory or directories in the order specified.

dbcdx.file.prep = *directory*

This property specifies the local directory where a file is created.

dbcdx.file.vol.*volume* **=** *directory* [**;** *directory* …]

This property specifies the local directory or directories associated with the *volume* that is specified. *volume* is case sensitive and must exactly match the volume specified on the open and prepare statements. This property may be specified more than once, except *volume* can not be duplicated.

DB/C Utilities

The DB/C Utilities are run from the operating system command prompt and from chain files. The Runtime Properties described in the previous chapter control the behavior of the utilities.

The syntax of the utilities and their parameters follows these rules:

1. All parameters are separated by blanks.

2. All parameters may be enclosed by or may contain a pair of double quotation marks (`" "`). The pair of double quotation marks is ignored. This technique may be used to enclose a blank character so that it will not be considered a delimiter between parameters. For example

   ```
   index file1 1-3 -p2=" "
   ```

   causes all records with blank characters in the second position to be selected, and

   ```
   index file1 1-3 -p2=A
   ```

   is exactly the same as

   ```
   index file1 1-3 -p2="A"
   ```

3. A solitary double quotation mark that is not part of a pair is invalid unless it is immediately preceded by the backslash character. For example

   ```
   index file1 1-3 -p2="
   ```

   will give an undefined result.

4. The back slash (`\`) character is the forcing character used when a parameter contains a double quotation mark or a backslash. The backslash forcing character is ignored. For example

   ```
   index file1 1-3 -p2=\\
   ```

   causes all records with a back slash in the second position to be selected.

5. Each utility is discussed in a separate chapter in this manual. At the beginning of each chapter, the command line syntax is presented. Certain parameters may be enclosed in brackets (`[ ]`). The brackets indicate that the enclosed parameter is optional.

6. If the utility is run as a stand-alone executable, each utility allows the **-cfg=***filename* command line parameter. This parameter provides the name of the runtime properties file. The runtime properties file contains information that controls various aspects of execution of each utility. If the **-cfg** parameter is not specified, then the **DBC_CFG** environment variable provides the runtime properties filename. If **-cfg** and **DBC_CFG** are not specified, then the default name for the runtime properties file is **dbcdx.cfg** which is to be found in the current directory. If the utility is run as an inline verb in the DB/C program, the **-cfg** command line parameter is not allowed.

aimdex

The aimdex utility creates a special aimdex file from a text file. This special aimdex file is used by the DB/C DX runtime for AFILE type files. The format of the aimdex command line is:

aimdex *file1* [*file2*] *key-spec* [*key-spec...*] [**-a=***n*] [**-cfg=***filename*] [**-d**] [**-e**] [**-f**[**=***n*]] [**-j**[**r**]] [**-m=***c*] [**-n=**[**+**]*n*] [**-o=***filename*] [**or**] [**-p***n*[**-***n*]**eq***c*[*c...*]] [**-p***n*[**-***n*]**ne***c*[*c...*]]] [**-p***n*[**-***n*]**gt***c*[*c...*]]] [**-p***n*[**-***n*]**ge***c*[*c...*]]] [**-p***n*[**-***n*]**lt***c*[*c...*]]] [**-p***n*[**-***n*]**le***c*[*c...*]] [**-r**] [**-s**] [**-t**] [**-x**[**=***n*]] [**-y**] [**-z=***n*]

file1 is the name of the input file. If no extension is specified, **.txt** is assumed. This file can be any text type file. The search path for the input file is controlled with the **dbcdx.file.open** runtime property.

file2 is the name of the output aimdex file. If *file2* is not specified, *file1* with an extension of **.aim** is assumed. If *file2* is specified without an extension, **.aim** is assumed. The search and create path for the output file is controlled with the **dbcdx.file.open** and **dbcdx.file.prep** runtime properties.

key-spec is a key specification. Valid key specifications are of the forms *nnn*, *nnn***-***nnn*, *nnn***x**, or *nnn***-***nnn***x**. *nnn* is the character position and *nnn***-***nnn* is the range of character positions from each input file text record that is used as a key. *nnn* is a number between 1 and 65520. The trailing letter **x** may be either upper or lower-case. The **x** signifies that the key information for this field is excluded from the AIM file, but the search criteria on read are still valid. Use of this parameter may improve read performance in certain cases and it may worsen read performance in other cases. Use of this parameter usually improves write performance.

-a=*n* is the memory allocation parameter where n is a one to three-digit number. *n* multiplied by 1024 is the number of bytes of memory that is allocated for the internal work area. This parameter affects only the performance of the aimdex command.

-cfg=*filename* is the runtime property file.

-d is the distinct key flag. If this parameter is specified, lower-case characters are distinct from upper-case characters. If this parameter is not specified, no case distinction is made.

-e is the existing command line parameters option. When this option is specified, the aimdex command utilizes the command line parameters that are stored in the existing aim file. If the existing aim file was created using the prep statement rather than the aimdex command, the command line parameters will not be present in the file and the **-e** option cannot be used. **-e** is mutually exclusive with all other command line parameters.

-f[**=***n*] is the fixed length record parameter. This parameter can only be specified if the records in the file are not compressed and are all the same length. The actual record size (*n*) only needs to be specified if the input file is initially empty. *n* is a number between 1 and 65500. If this parameter is not specified, variable length and compressed records may be used. Read performance is usually improved if this parameter is specified. Write performance is not affected by this parameter.

-j[**r**] is the share open mode parameter. The **-j** parameter causes the input file to be opened in share read/write mode. The **-jr** parameter causes the input file to be opened in share read-only mode.

-m=*c* is the match character parameter. *c* is the match character that is compared to all characters where it is specified in the read key pattern. If not specified, the match character defaults to **?**.

-n=[**+**]*n* is the primary extent number of records parameter. *n* is a one to seven-digit number. This parameter specifies the amount of space to be allocated in the primary key area of the aim file. If the typical maximum number of records in the text file is known, this parameter should be specified. Both read and write access perform better if the aim file does not grow into secondary extents. If the optional **+** is specified, then the amount of space reserved in the primary key area of the AIM file is the sum of the amount of space required for the number of records currently in the input file plus the amount of space required for *n* additional records.

-o=*filename* is the options file parameter. *filename* is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

or is the logical or parameter. If the **or** parameter is placed between two **-p** parameters, they are logically OR-ed together instead of logically AND-ed together. The precedence of all **-p** operations is from left to right.

-p*n*[**-***n*]**eq***c*[*c*...]
-p*n*[**-***n*]**ne***c*[*c*...]
-p*n*[**-***n*]**gt***c*[*c*...]
-p*n*[**-***n*]**ge***c*[*c*...]
-p*n*[**-***n*]**lt***c*[*c*...]
-p*n*[**-***n*]**le***c*[*c*...] are the record select range parameters. *n* may be a one to four digit decimal number. *c* is the match character. The key of a record is included in the aimdex if the record contains a character that is equal, not equal, greater than, greater than or equal, less than, or less than or equal to (respectively) the match character at character position *n*. Otherwise, the record is ignored. **=** may be used instead of **eq**. **#** may be used instead of **ne**. The optional *n*-*n* is used to specify a range of character positions. If a range is specified, the number of match characters specified should be the same as the number of positions in the range. If the number of match characters specified is less than the number of positions in the range, it is assumed that the unspecified match characters are blanks. If the number of match characters specified is greater than the number of positions in the range, the extra match characters are truncated. If more than one **-p** parameter is specified, they are logically AND-ed together. For example:

-p5eqxy

causes the records with a match character at the fifth character position equal to **x** or equal to **y** to be selected. If this parameter is specified, the **-f** parameter must also be specified.

-r is the rename file parameter. This parameter causes the name of the text file stored in the AIM file to be changed to the specified input text file name. This parameter is mutually exclusive with all other parameters. Both the input text file and the output aim file must be specified.

-s is the space reclamation parameter. This parameter is only applicable if the **-f** parameter is also specified. If the **-s** parameter is specified, records written to the file reuse the space made available by deleted records whenever possible.

-t is the limited text file search parameter. If this parameter is specified, the open statement will only look for the text file in the same directory in which the index file is found.

-x[**=***n*] without =*n* is the reverse selection parameter. This parameter causes those records that are selected by the **-p** parameter to be excluded from the output file and those records not selected to be written into the output file. **-x** with =*n* specified is the secondary extent number of records parameter. *n* is a one to seven-digit number. This parameter specifies the amount of space that is allocated when the primary key area is full. If not specified, each secondary extent contains space for 25% of the number of records in the primary area.

-y is the test end of file parameter. This parameter causes aimdex to fail if an end of file character is encountered other than at the physical end of file.

-z=*n* is the speed parameter. If not specified, the default Z value is 199. Valid Z values are from 40 to 2000. As the Z value increases, read performance improves. However, the size of the aim file also increases linearly with the Z value. Write performance typically improves as the Z value decreases.

The aim search can be case sensitive or case insensitive, depending if the **dbcdx.file.casemap** runtime property is specified.

The aimdex utility returns a return code of zero if aimdex completes successfully. The return code is non-zero if aimdex does not complete successfully. If the return code is non-zero and aimdex is running under chain execution, chain execution is aborted.

build

The build utility places one or more records selected from an input file into an output file. The format of the build command line is:

build *file1 file2* [*rec-spec...*] [**-a**] [**-c**] [**-cfg=**filename] [**-j**[**r**]] [**-***n=filename*] [**-o=***filename*] [**or**] [**-p***n*[**-***n*]**eq***c*[*c...*]] [**-p***n*[**-***n*]**ne***c*[*c...*]]] [**-p***n*[**-***n*]**gt***c*[*c...*]]] [**-p***n*[**-***n*]**ge***c*[*c...*]]] [**-p***n*[**-***n*]**lt***c*[*c...*]]] [**-p***n*[**-***n*]**le***c*[*c...*]] [**-s=***filename*] [**-t**[**=***type*]] [**-x**] [**-y**]

file1 is the name of the input file. If no extension is specified, then **.txt** is assumed. This file can be any text type file. The name of the input file and the name of the output file must be different. The search path for the input file is controlled with the **dbcdx.file.open** runtime property.

file2 is the name of the output file. If no extension is specified then **.txt** is assumed. The name of the input file and the output file must be different. The search and create path for the output file is controlled with the **dbcdx.file.open** and **dbcdx.file.prep** runtime properties.

rec-spec is a record specification. Valid record specifications are of the form *nnn* or *nnn***-***nnn*, where *nnn* is a record number to be included in the output and *nnn***-***nnn* is a range of records to be included in the output. If a record specification is not designated, all records are included in the output.

-a appends the records from the input file to the output file. This implies that the output file must already exist. If the **-a** parameter is not specified, then the records are placed into the output file as the first records, overwriting any records that already exist. If this parameter is specified, then the output file type is not changed, and the **-t** parameter is ignored if it is specified.

-c causes the output file to be in DB/C text compressed format. The **-c** and **-t** parameters are mutually exclusive.

-cfg=*filename* is the runtime property file.

-j[**r**] is the share open mode parameter. The **-j** parameter causes the input file to be opened in share read/write mode. The **-jr** parameter causes the input file to be opened in share read-only mode.

-n=*filename* is the read translate table. filename contains the 256 byte translate table applied to each record as it is read.

-o=*filename* is the options file parameter. *filename* is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

or is the logical or parameter. If the **or** parameter is placed between two **-p** parameters, they are logically OR-ed together instead of logically AND-ed together. The precedence of all **-p** operations is from left to right.

-p*n*[**-***n*]**eq***c*[*c...*]
-p*n*[**-***n*]**ne***c*[*c...*]
-p*n*[**-***n*]**gt***c*[*c...*]
-p*n*[**-***n*]**ge***c*[*c...*]
-p*n*[**-***n*]**lt***c*[*c...*]
-p*n*[**-***n*]**le***c*[*c...*] are the record select range parameters. *n* may be a one to four digit decimal number. *c* is the match character. The key of a record is included in the aimdex if the record contains a character that is equal, not equal, greater than, greater than or equal, less than, or less than or equal to (respectively) the match character at character position *n*. Otherwise, the record is ignored. **=** may be used instead of **eq**. **#** may be used instead of **ne**. The optional *n***-***n* is used to specify a range of character positions. If a range is specified, the number of match characters specified should be the same as the number of positions in the range. If the number of match characters specified is less than the number of positions in the range, it is assumed that the unspecified match characters are blanks. If the number of match characters specified is greater than the number of positions in the range, the extra match characters are truncated. If more than one **-p** parameter is specified, they are logically AND-ed together.

-s=*filename* is the unselected records parameter. filename is the name of the file to write all records not selected with the **-p** parameter. The file specified by filename is always created by build and the **-a** parameter has no impact on the creation of this file.

-t[**=***type*] is the output file type option. *type* can be **data**, **crlf**, **text**, or **std**. The default is to create DB/C standard text file (*type* is **std**). Under Windows, **text** is the equivalent to **crlf**; otherwise, **text** is the equivalent to **data**. If *type* is omitted, **text** is assumed.

-x is the reverse selection parameter. This parameter causes those records that are selected by the **-p** parameter to be excluded from the output file and those records not selected to be written into the output file.

-y is the test end of file parameter. This parameter causes build to fail if an end of file character is encountered other than at the physical end of file.

The build utility returns a return code of zero if build completes successfully. The return code is non-zero if build does not complete successfully. If the return code is non-zero and build is running under chain execution, then chain execution is aborted.

chain

The chain utility provides the ability to execute a series of commands without operator intervention. The actual commands executed may be controlled by parameters specified at chain runtime. To run a series of commands, a chain file is created which contains the commands and chain directives. This chain file is then executed by running the chain command. The format of the chain command line is:

chain *file* [*parameter*] [**-c**] [**-cfg=***filename*] [**-d**] [**-i**] [**-j**[**r**]] [**-n**] [**-o=***filename*] [**-p**] [**-s**] [**-t**] [**-***w=filename*]

file is the user-created chain file. If no extension is specified, **.chn** is assumed. This file can be any text type file. A file name is required unless **-n** or **-p** is specified. The search path for the input file is controlled with the **dbcdx.file.open** runtime property.

parameter is a chain variable. A chain variable must have a valid variable name. Each chain variable specified as a *parameter* is set to the logical value true, character variable type, and string value null. If the variable is followed by an equal sign and a string of characters, the string value of the variable is set to the value of the string of characters following the equal sign. One or more *parameters* may be specified.

-c causes the chain command to compile the chain file, but not to execute the resulting commands.

-cfg=*filename* is the runtime property file.

-d causes the chain command to display each chain file input line after the replacement of symbols.

-i is the case insensitive parameter. By default, chain variables are case sensitive. The **-i** parameter causes them to be case insensitive.

-j[**r**] is the shared open mode parameter. The **-j** parameter causes the input file to be opened in shared read/write mode. The **-jr** parameter causes the input file to be opened in shared read-only mode.

-n is a restart parameter that causes chain execution to continue from a previously aborted chain execution. The **-n** parameter causes execution to continue at the next command after the one that was aborted.

-o=*filename* is the options file parameter. *filename* is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

-p is a restart parameter that causes chain execution to continue from a previously aborted chain execution. The **-p** parameter causes execution to continue at the previously executing command (the one that was aborted).

-s causes the chain execution phase to use the system command interpreter or shell to execute the commands. After the execution of a command, the abort bit will be cleared. The return status from the executed command will be ignored. The **-s** option enables the user to take advantage of some of the features of the command interpreter such as input and output redirection, batch or script file support, pipe support, and multi-tasking support. The use of the **-s** option may cause the output of the executed commands to ignore the redirection of the log directive. The systemoff directive cancels the effect of the **-s** option.

-t causes the debug directive to display the variables on the control statement. Otherwise, the debug directive is ignored.

-w=*filename* is the work file parameter. If not specified, the default work file name is **chain.wrk**.

The chain utility returns a return code of zero if chain completes successfully. The return code is non-zero if chain does not complete successfully (i.e., chain aborted).

Chain executes in two phases. The first phase is the compilation phase. The second phase is the execution phase.

The compilation phase creates the chain work file and writes an extent that contains command lines and execution time directives.

The execution phase reads the work file and performs the commands and execution time directives. When a command is executed during the execution phase, the command returns a status value. If that value is non-zero, the execution is considered a failure and the abort bit of chain is set. If the return value is zero, the execution is considered a success and the abort bit of chain is cleared. Immediately after the command execution, the abort bit is tested. If the bit is set, chain execution is aborted and chain returns a status value of one. The noabort directive alters the testing of the abort bit. The systemon directive and the **-s** option force the abort bit to be cleared, even if the return value was non-zero (the command failed). This feature is useful when executing non-DB/C commands, or when you want to ignore errors.

If chain itself is executed by the execution phase, chain is being executed recursively. In this case, the compilation phase appends a secondary extent to the work file. chain execution then continues with the first command in the new extent. When all commands in that extent have been executed, the extent is removed and chain continues with the remainder of the primary or current extent. There is no limit on the number of times chain may be called recursively. An aborted chain that was called recursively will abort the original chain .

Replacement Characters

Chain input lines may contain variables that are replaced by a string of characters during the compilation phase. The use of the pound (**#**) character directly before and after a variable name in a chain line causes the pound signs and the variable name to be replaced by the string value associated with the variable name. However, no replacement takes place if the logical value of the variable is false or if the logical value is true and the string value is null. The effect is to remove the pound signs and the variable name from the statement.

The use of the percent (%) character is similar to that of the pound sign except that the string value of the variable must be numeric. That value is then converted to a string of octal digits before the replacement takes place. The first digit of the octal string is always 0.

Nested replacements are supported.

Chain Statements and Chain Command Lines

All lines in the chain file are either chain statements or command lines. Lines that begin with **//**, **/@**, **/&**, **/.**, **/:**, or /* are chain statements. Any chain line that is not a chain statement is considered to be a chain command line.

A chain command line is written to the current extent of the chain work file and will be executed during the execution phase of chain. chain command lines may be altered by the chain statements.

The chain statements are:

// *directive* is the chain compile time control statement. These lines must contain a directive and zero or more directive operands. The chain directives are described later in this chapter.

/@ *directive* is the chain compiler "as is" control statement. It is the same as the **//** compile time control statement, except no replacements are performed.

/& *comment* is the pure comment statement. Lines starting with **/&** will never be displayed and will have no effect on compilation or execution.

/. *comment* is the compile time comment statement. Lines starting with **/.** will be displayed during the chain compilation phase.

/: *comment* is the execution time comment statement. Lines starting with **/:** will be displayed during the chain execution phase. NOTE: If the **/:** is immediately followed by **B** or **C**, a short beep will be sounded. To include a comment that begins with the letter **B** or the letter **C**, place a space immediately after the **/:** .

/* *comment* is the execution time pause comment statement. It is the same as the **/:** execution time comment statement, except execution is halted until the enter key is pressed. NOTE: If the

/* is immediately followed by **B** or **C**, a long beep will be sounded. To include a comment that begins with the letter **B** or the letter **C**, place a space immediately after the /* .

Chain Variables

Each chain variable contains three attributes. The attributes are:

logical value	true or false
string value	a string of characters
open file association	true or false

Variable names may only contain alphanumeric characters. The first character must be alphabetic. Variable names may be up to 12 characters in length. Variable names longer than 12 will be truncated to 12 characters by using the first 11 characters and the last character. Variable names greater than 12 should be verified for uniqueness. Variable names are case sensitive unless the **-i** parameter is specified.

Variables are created in two ways. First, a variable may be created from the list of variable names in the command line. Second, a variable may be created by being used in a chain directive. The following directives can be used to create variables or to change the values of existing variables: **//assign**, **//keyin**, **//open**, **//openread**, **//read**, and **//set**.

The maximum length of the string value is 80 characters. The logical value may be true or false. If the logical value is false, then the string value is considered null.

When a variable is associated with an open file by using the **//open** or **//openread** directives, the string value is changed to the name of the file and the open file association is set to disallow modification of the string value by **//assign**, **//set**, or **//keyin**.

There are several reserved variable names. In all cases, their string value is the null string. Their logical values are:

Name	Logical Value
FALSE or **false**	false
No or **no**	false
OFF or **off**	false
TRUE or **true**	true
YES or **yes**	true
ON or **or**	true
NULL or **null**	true

Chain Literals

Chain literals are used with control statement directives. A literal may be used anywhere a variable may be used, except as the object of an assignment with directives that create variables.

A literal is enclosed in double quotation marks (**""**). To include a quotation mark (**"**) character in a character literal, use the pound (**#**) character as a forcing character. To include a pound character in a character literal, use two pound characters in a row. A numeric literal does not need to be enclosed in double quotation marks.

Chain Expressions

Chain expressions consist of variables, literals, and operators. chain supports the following operators: arithmetic, logical, character string, and special functions. Expressions are evaluated from left to right and no operator has precedence over another. Parentheses can be used around an operation or a group of operations to force precedence in evaluation. Parentheses may be nested.

Chain Directives

Directives are reserved key words that control the operations of the chain command. The compile time control statements require a chain directive.

There are two types of chain directives. There are compile time directives and execution time directives. The compile time directives control the operations during the compilation phase of chain. The execute time directives control the operations during the execution phase of chain.

Compile time directives are: **abort**, **assign**, **beep**, **click**, **close**, **debug**, **discard**, **display**, **do**, **else**, **elseif**, **end**, **if**, **include**, **keyin**, **open**, **openread**, **position**, **read**, **rewrite**, **set**, **sound**, **stop**, **until**, **wait**, **while**, **write**, and **xif**.

Execute time directives are: **abortoff**, **aborton**, **abtif**, **errabort**, **errgoto**, **goto**, **keyboard**, **label**, **log**, **logoff**, **noabort**, **soundr**, **stamp**, **systemoff**, **systemon**, and **terminate**.

// abort
The abort compiler directive ends compilation and stops any execution of chain commands compiled so far. Any nested levels of chain are also aborted.

// abortoff
The abortoff execution directive clears the chain abort bit.

// aborton
The aborton execution directive sets the chain abort bit.

// abtif
The abtif execution directive aborts chain execution if the chain abort bit is set.

// assign *variable-name* = *operand*
variable-name is a variable
operand is a variable, literal, or expression

The assign directive is a string manipulation compiler directive. It changes the value of an existing variable or creates a new variable. If variable-name does not exist, it is created. The string value of the variable-name is changed to the string value of operand. The logical value of variable-name is set to true.

// beep
The beep compiler directive sounds a beep during compilation.

// click
The click compiler directive sounds a beep during compilation.

// close *variable-name*
variable-name is a variable

The close compiler directive closes the file associated with the variable specified by *variable-name*. If the variable is not associated with an open file, nothing happens.

// debug *list*
list is a list of operands that are variables and expressions separated by blanks

The debug compiler directive helps the user debug the chain file. The value of the variables and expressions specified by the operands will be displayed separated by a space. The debug directive will be ignored unless the **-t** option was specified on the chain command line.

// discard *variable-name*
variable-name is a variable

The discard compiler directive removes the variable specified by *variable-name*. If the variable is associated with an open file, the file is closed before the variable is removed.

// display *list*
list is a list of operands that are variables and literals separated by blanks

The display directive is a compiler directive that displays the items from the list. If the operand in the *list* is a variable, its string value is displayed. If the operand in the list is a literal, its value is displayed.

// do

// until *operand*

operand is a variable or expression

The do and until directives are compiler directives which conditionally control execution of subsequent command lines and compilation of the subsequent chain directives.

If the logical value of the variable or expression following the until directive is false, compilation continues at the line containing the do directive. If the logical value of the variable or expression is true, compilation continues at the chain line following the line containing the until directive.

The chain lines between the line containing the do directive and the line containing the until directive will be processed at least one time. The do directive and the matching until directive must be contained within the same chain file.

// errabort

The errabort execution directive causes chain to test the abort bit after a command line is executed. If the abort bit is set, the chain is aborted. This is the default setting at the start of chain, the start of a nested chain, and after an executed command sets the abort bit. This directive is canceled by the noabort execution directive.

// errgoto *label*

label is a label name

The errgoto execution directive causes execution to continue at the line after the specified label if the abort bit is set.

// goto *label*

label is a label name

The goto execution directive causes execution to continue at the line following the specified label.

// if *directive-operand*
// elseif *directive-operand*
// else
// xif

directive-operand is a variable or expression

The if, elseif, else, and xif compiler directives conditionally control execution of subsequent command lines and compilation of subsequent chain directives.

If the logical value of the variable or expression following an if directive is true, compilation continues at the next chain line. If the logical value of the variable or expression is false, then any elseif directives are evaluated. If the logical value of any of the elseif directives is true, compilation continues at the next chain line. If the logical values of all the if and elseif directives are false, compilation continues after the optional else directive if it exists, or after the required xif if the else does not exist.

// include *operand*

operand is a chain file

The include compiler directive causes chain compilation to continue with statements from another chain file. The *operand* must be a valid chain file name. If no extension is specified, **.chn** is assumed. Includes may be nested up to three levels deep.

// keyboard *operand*

operand is a variable or literal

The keyboard interactive execution directive allows the user to display a variable or literal specified by the operand. The user is prompted for a response. If the response is null (i.e., the Enter key is pressed), execution continues. If a string of characters is entered, that string is treated as a command line and is executed immediately.

// keyin *list*

list is a list of variables and literals separated by blanks

The keyin interactive compiler directive allows the user to input data from the keyboard and display data on the screen. The use of a variable in the list causes that variable to be created or modified. A maximum of 80 characters may be entered as the string value of the variable. The use of a literal in the list causes the value of the literal to be displayed.

// label *label*
label is the label name

The label directive is an execution directive. The directive operand following the label directive defines a place in the chain file that can later be accessed by an errgoto or goto directive.

// log *operand*
operand is a log file

The log execution directive causes the output from chain and the output from commands executed from chain to be redirected to the file specified by the operand. Output will not be displayed on the screen. If the file already exists, the output will be appended to the end of file. The log directive may not be supported for all operating systems and the results are unknown if chain is executed with the **-s** option or the systemon directive in effect.

// logoff
The logoff execution directive cancels the log directive and causes chain and command output to be directed to the screen.

// noabort
The noabort execution directive cancels the errabort execution directive and prevents the chain from aborting after an executed command line sets the abort bit. The abort bit will remain set after an executed command line sets it. The noabort is canceled by the errabort execution directive. The noabort is also canceled by an implicit errabort, which occurs immediately after an executed command sets the abort bit or the start of a nested chain.

// open *operand1 operand2*
operand1 is a file variable
operand2 is the file to open

The open compiler directive opens the file specified by *operand2* and assigns the information to the variable specified by *operand1*. The file is opened in exclusive read/write mode. If the file does not exist, the file is created. If the open fails, chain compilation is aborted with an error message. If the variable specified by *operand1* is already associated with an open file, then that file is closed before the open is performed.

// openread *operand1 operand2*
operand1 is a file variable
operand2 is the file to open

The openread compiler directive is the same as the open directive, except the file is opened in shared read-only mode.

// position *operand1 operand2*
operand1 is a variable associated with an open file
operand2 is a variable or literal

The position compiler directive sets the current position of the file specified by *operand1* to the position specified by *operand2*. An attempt to position beyond the end of file will result in a warning message.

// read *operand list*
operand is a variable associated with an open file
list is a list of variables separated by blanks

The read compiler directive reads the record located at the current position of the file specified by the operand. The characters in the record are moved into the variables in the list. Up to 80

characters are moved into a single variable. If there are less than 80 characters of data, only that number of characters will be moved to the variable.

For the first variable in the *list*, the logical value will be set to false if the end of file is read; otherwise, the logical value will be set to true. If a record with length zero is read, the string value will be set to null.

For the remaining variables in the list, if there was no data to move, then the logical value is set to false. Otherwise, data is moved and the logical values are set to true. After the read, the current position of the file is advanced to the next record.

// rewrite *operand list*
operand is a file variable
list is a list of variables and literals separated by blanks

The rewrite compiler directive writes a record in the file specified by the operand. The record is written at the position of the last record accessed, unless modified by the position directive. The data written is from the list of variables or literals. The length of the record at the current position determines the maximum number of bytes written. After the write, the current position of the file is set to the first byte of the next record.

// set *variable-name* **=** *operand*
variable-name is a variable
operand is a variable, literal, or expression

The set compiler directive is used to change the logical value of an existing variable or to create a new variable. If the variable specified by *variable-name* does not exist, it is created. The logical value of *variable-name* is changed to the logical value of operand. The string value of *variable-name* is set to null.

// sound *operand*
operand is an optional variable or literal

The sound compiler directive causes a beep to sound for the number of seconds specified by the operand. The beep will last for a minimum of one second.

// soundr *operand*
operand is an optional variable or literal

The soundr execution directive causes a beep to sound for the number of seconds specified by the operand. The beep will last for a minimum of one second.

// stamp
The stamp execution directive displays the current time and date.

// stop *operand*
operand is an optional variable or literal

The stop compiler directive stops chain compilation immediately. The optional operand is a command line that will be executed at the start of the execution phase.

// systemon
Specifying the systemon execution directive is the same as specifying the **-s** option on the chain command line. This will cause all the following commands to be executed with the system command interpreter or shell. After a command is executed, the chain abort bit will be cleared. The return status from the command execu ed will be ignored.

The systemon directive enables the user to take advantage of some of the features of the command interpreter such as input and output redirection, batch or script file support, pipe support, and multi-tasking support. The use of the systemon directive may cause the output from the executed commands to ignore the redirection of the log directive. The systemoff directive cancels the systemon directive.

// systemoff
The systemoff execution directive cancels the effects of the systemon directive or the **-s** option on the command line.

// **terminate**

The terminate execution directive terminates chain execution for the current chain extent. Other extents will continue to execute.

// **wait** *operand*

operand is an optional variable or literal

The wait compiler directive causes chain compilation to pause for the number of seconds specified by the *operand*. wait will pause for a minimum of one second.

// **while** *operand*
// **end**

operand is a variable or expression

The while and end compiler directives conditionally control execution of subsequent command lines and compilation of subsequent chain directives.

If the logical value of the variable or expression following the while directive is true, compilation continues at the next chain line and when the end directive is encountered, compilation continues with the while directive. If the logical value of the variable or expression following the while directive is false, compilation continues at the chain line following the line containing the end directive.

// **write** *operand list*

operand is a variable associated with an open file
list is a list of variables and literals separated by blanks

The write compiler directive writes a record at the end of the file specified by the operand. The data written is from the list of variables or literals. After the write, the current position of the file is set to the end of the file.

Arithmetic Operators

The arithmetic operators are: addition, subtraction, multiplication, division, and modulus.

+	addition
–	subtraction or negate
*****	multiplication
/	divide
//	modulus

An arithmetic operator is placed between two variables and/or literals except when the minus sign is used to negate a single variable or literal. These operations are performed on decimal numeric values. The following paragraphs describe the methods by which the numeric values are obtained.

If a variable in the operation has a logical value of false, then that variable assumes a numeric value of zero.

If the variable in the operation has a logical value of true, then the string value is converted to a number before the arithmetic operation occurs. If the string value is not a valid numeric string, its numeric value is zero.

The result of an arithmetic operation is a string value that does not contain leading zeros or blanks. The resulting logical value is true.

Logical Operators

These are the logical operators:

~	unary not
&	and
|	or
=	equal

~=	not equal
<	less than
>	greater than
<=	less than or equal
>=	greater than or equal

A logical operator is placed between two variables and/or literals except for unary not which operates on a single variable or literal and is placed to the left of the variable or literal. The unary not, and, and or operators are performed on the logical value of the variables. The other operators operate on the string values of the variables.

If a variable in the operation has a logical value of false, then that variable assumes a string value of null (**""**). If the variable in the operation has a logical value of true, different string values can be assumed.

The result of a logical operation is a logical value of true or false and a string value of null.

Character String Operators

The character string operators are: concatenation, substring, substring control, string length, pattern match, string scan, string scan unequal, and string replace.

****	concatenation
^	substring
:	substring control
.	string length
[	pattern match
[[	string scan
~[[	string scan unequal
****	string replace

A character string operator is placed between two variables and/or literals except for the substring operator which has the syntax described below. These operations are performed on character string values. The character string operators use the same methods as the logical operators to obtain the string values.

The concatenation operation creates a new string by appending one string to another. The format is:

varlit1 **** *varlit2*
varlit1 is the first string
varlit2 is the second string

The result is a value that has the type of character. The logical value is true. The string value is the second operand appended to the first, but this value may not exceed 80 characters.

The substring operation extracts the specified series of characters from an existing string. The formats are:

varlit **^** *exp1*
varlit **^ (***exp1* **:** *exp2***)**
varlit is a variable or literal from which the characters are extracted
exp1 is the start expression which can be a variable, literal, or arithmetic expression
exp2 is the length expression which can be a variable, literal, or arithmetic expression

In the first format, the character in position *exp1* of the string value associated with *varlit* is extracted. In the second format, *exp2* characters starting at position *exp1* of the string value associated with *varlit* are extracted. If the value of *exp1* is negative, then the position is calculated from the end of the string. If *exp1* is zero or the absolute value is greater than the length of the string, no characters are extracted. If *exp2* is negative, then the characters are extracted in reverse order. If *exp2* is specified and the end or beginning of string is reached before the all characters are extracted, then the string is considered circular and the extraction continues at the opposite end.

The string value is the characters extracted, but this value may not exceed 80 characters. The logical value result is true.

The string length operation determines the length of a string. The format is:

.*varlit*
varlit is a variable or literal

The result is a string value that is the length of the string value of *varlit*. The logical value result is true.

The pattern match operation finds the first occurrence of a substring within a string. The format is:

varlit1 **[** *varlit2*
varlit1 is the string to be searched
varlit2 is the search string

The string value of *varlit2* may contain the character **?**, which is considered the match character.

If the string value of *varlit2* was not found in the string value of *varlit1*, the logical value result is false. If the substring was found, the logical value is true and the resulting string value is the starting position within *varlit1* of the matched substring.

The scan operation finds the first occurrence of any characters in a string that are contained in a search string. The format is:

varlit1 **[[** *varlit2*
varlit1 is the string to be searched
varlit2 is the search string

If the characters within the string value of *varlit2* are not found in the string value of *varlit1*, the logical value result is false. If any of the characters in string *varlit2* are found in *varlit1*, the logical value is true and the resulting string value is the position within *varlit1* of the first character that matched.

The scan unequal operation finds the first occurrence of any characters in a string that are not contained in a search string. The format is:

varlit1 **~ [[** *varlit2*
varlit1 is the variable to be searched
varlit2 is the search string

If the characters within the string value of *varlit2* are found in every position of the string *varlit1*, the logical value result is false. If the characters in the string value of *varlit2* are not found in *varlit1*, the logical value is true and the resulting string value is the position within *varlit1* of the first character that could not be matched.

The string replace operation creates a new string value by replacing selected characters within a string with other characters. The format is:

varlit1 **** *varlit2*
varlit1 is the string where the replacement takes place
varlit2 is the replacement string

The string value of *varlit2* contains pairs of characters. The first character of a pair is the character to replace. The second character is the character that will replace the first. If the length of the string value *varlit2* is not an even number, then no replacement will take place.

The logical value result is true. The resulting string value is the *varlit1* string value after replacement has occurred.

Special Function Operators

The function operators are: **ffile**, **getpos**, **getlastpos**, and **clock**. The function operators operate on a single variable except for the clock operator which does not operate on any variables.

The ffile operation determines if a file is available. The format is:

ffile (*varlit***)**
varlit is a variable or literal that contains the name of the file.

If the file could not be found, the logical value result is false. If the file is found, the logical value result is true. If the file can be opened in exclusive read/write mode, the value of the string is null (length of zero);

otherwise the value is a string that contains an error message and the length is the length of the error message.

The getpos operation returns the current file position associated with a file. The format is:

getpos (*var***)**
var is a variable that is associated with an open file

If the variable is not associated with an open file, then the logical value result is false. If it is associated with a file that is open, the logical value is true and the resulting string value is the current file position.

The getlastpos operation returns the file position of the most recent access to a file. The format is:

getlastpos (*var***)**
var is a file variable

If the variable is not associated with an open file, then the logical value result is false. If it is associated with a file that is open, the logical value is true and the resulting string value is the previous file position.

The clock operation returns the current date and time. The format is:

clock ()

The value is the string that contains the date and time in the format:

*yyyy***/***mm***/***dd hh***:***mm***:***ss*. The length is set to 19. The logical value result is true.

copy

The copy utility copies an input file to an output file on another disk. The format of the copy command line is:

copy *file1 file2* [**-cfg=***filename*] [**-d**] [**-j**[**r**]] [**-o=***filename*]

file1 is the name of the input file. If no extension is specified, **.txt** is assumed. The search path for the source file is controlled with the **dbcdx.file.open** runtime property.

file2 is the name of the output file. If no extension is specified, **.txt** is assumed. The search and create path for the output file is controlled with the **dbcdx.file.open** and **dbcdx.file.prep** runtime properties.

-cfg=*filename* is the runtime property file.

-d is the delete input file parameter. If the copy command is successful, then the input file is deleted.

-j[**r**] is the share open mode parameter. The **-j** parameter causes the input file to be opened in share read/write mode. The **-jr** parameter causes the input file to be opened in share read-only mode.

-o=*filename* is the option file parameter. *filename* is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

The copy utility returns a return code of zero if copy completes successfully. The return code is non-zero if copy does not complete successfully. If the return code is non-zero and copy is running under chain execution, then chain execution is aborted.

create

The create utility creates a file. If the designated file already exists, it is truncated, effectively erasing any existing

records. The format of the create command line is:

`create` *file1* [**`-cfg=`***filename*] [**`-l=`***n*] [**`-n=`***n*] [**`-o=`***filename*] [**`-t`**[**`=`***type*]]

file1 is the name of the file to be created. If no extension is specified, **.txt** is assumed. The search and create path for the output file is controlled with the **dbcdx.file.open** and **dbcdx.file.prep** runtime properties.

`-cfg=`*filename* is the runtime property file.

`-l=`*n* is the length parameter. *n* is the length of the records. *n* is a number between 1 and 8192. This parameter must be used in conjunction with the **-n** parameter to create a file with the specified number of length *n* blank records.

`-n=n` is the number of records parameter. This parameter must be used in conjunction with the **-l** parameter to create a file with the specified number of length *n* blank records.

`-o=`*filename* is the options file parameter. filename is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

-t[**=***type*] is the output file type option. *type* can be **data**, **crlf**, **text**, or std. The default is to create DB/C standard text file (**std**). Under Windows, **text** is the equivalent to **crlf**; otherwise, **text** is the equivalent to **data**. If type is omitted, **text** is assumed.

The create utility returns a return code of zero if create completes successfully. The return code is non-zero if create does not complete successfully. If the return code is non-zero and create is running under chain execution, then chain execution is aborted.

delete

The delete utility deletes the specified file. The format of the delete command line is:

`delete` *file1* [**`-cfg=`***filename*]

file1 is the name of the file to be deleted. If no extension is specified, **.txt** is assumed. The search path for the file is controlled with the **dbcdx.file.open** runtime property.

`-cfg=`*filename* is the runtime property file.

The delete utility returns a return code of zero if *file1* exists and is successfully deleted. The command also returns a return code of zero if *file1* could not be found and will display a message indicating it. The command returns a nonzero return code if *file1* exists and cannot be opened or is not specified. If the return code is non-zero and delete is running under chain execution, the chain execution is aborted.

dump

The dump utility allows examination and alteration of any file. The format of the dump command line is:

dump [*file1*] [**-b**[=*n*]] [**-cfg=***filename*] [**-j**[**r**]] [**-o=***filename*] [**-v**]

file1 is the name of the file to be examined and possibly altered. If no extension is specified, **.txt** is assumed. The search path for the input file is controlled with the **dbcdx.file.open** runtime property.

-b[**=***n*] is the block size parameter. *n* is a number between 1 and 8192. It is the number of bytes that are read from the file. If *n* is not specified, the default is 400 bytes. If **-b** is not specified, the block size is set to the size of the first record in the file.

-cfg=*filename* is the runtime property file

-j[**r**] is the share open mode parameter. The **-j** parameter causes the input file to be opened in share read/write mode. The **-jr** parameter causes the input file to be opened in share read-only mode.

-o=*filename* is the options file parameter. *filename* is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

-v is the ignore **dbcdx.display** and **dbcdx.keyin** runtime properties.

When the dump command is invoked, the prompt **CMD:** is displayed. The user may enter commands at this prompt. The commands available are:

b	change block size
c	Change to character mode
d	decrease the current block pointer by one and display that block
e	go to the end of file
f	fill all bytes in the current block with fill character
h	change to hex mode
i	increment the current block pointer and display that block
k	read by block number
l	change fill character
m	modify bytes in the current block
n	specify a new file to dump
o	read by position
p	show another page of the block (for blocks > 400 bytes)
q	quit and exit to operating system
r	read in a block of data and display it
s	search the file for occurrence of a string
u	decompress a portion of the current block
w	write the current block
?	display this screen

The dump commands allow a block of data to be read from the file, displayed or altered, and then written back to the disk. The default block size is the length of the first record. The block size may be changed with the **-b** command line parameter or the **b** dump command. The mechanism for specifying the data to be read from the file is either by block number or by file position. This may also be changed with the **k** and **o** commands.

When in hexadecimal mode for a search or modify or when entering the fill character, entering **#***dd* (where *d* is a digit or period) causes the two characters after the **#** to be converted to a single byte that is the compressed representation of those two characters.

edit

edit is a full-screen source code editor. The format of the edit command line is:

edit [*file1*] [**-a=***n*] [**-cfg=***filename*] [**-t=***type*] [**-v**]

file1 is the name of the file to be edited. If no extension is specified, **.txt** is assumed. If the file does not exist, it is created. This file can be any text type file. The search path is controlled with the dbcdx.file.source runtime property.

-a=*n* is the memory allocation parameter. The **-a** parameter allows the user to specify the amount of memory available for files that are being edited. *n* is the amount of memory in kilobytes.

-cfg=*filename* is the runtime property file.

-*t=type* is the output file type option. type can be **data**, **crlf**, **text**, or **std**. If the file does not exist, this option can control the type of file created. The default is to create **text** type text file. Under Windows, **text** is the equivalent to **crlf**; otherwise, **text** is the equivalent to **data**.

-v is the ignore **dbcdx.display** and **dbcdx.keyin** runtime properties.

Configuration

All editor functions and commands may be configured for different keyboards and operating environments. The Set Options command in the editor allows the user to alter key settings. The default keystroke for the Set Options command is **Ctrl-O**. The option settings screen shows all available functions and commands.

More information about the option settings screen is provided later in the chapter.

Overview of the Editor

When the edit command is invoked, the edit screen is displayed.

The top line of the screen is the response line. This line displays pertinent information. It also prompts the user for input during execution of certain commands. The file name, the current line number, the total number of lines in the file, and the current buffer are displayed at the far right of the response line.

A horizontal line is displayed on the second line of the screen.

All lines between the third line and the last line on the screen are used to display the file being edited.

Commands

Exit

Default key: **Ctrl-X**

This command is used to exit the editor. If the current file has been modified since the last time it was written to disk, then the message **OK to lose changes?** is displayed on the response line. If the user types **Y** or **y**, then all the changes made since the last time the file was written to disk are lost, and control returns to the operating system. If the user types any other letter, then nothing occurs.

Move Cursor Left

Default key: **LEFT**

This command moves the cursor one position to the left. If the current line is longer than the width of the screen and the cursor is positioned at the far left position of the screen, then the screen is scrolled.

Move Cursor Right

Default key: **RIGHT**

This command moves the cursor one position to the right. If the current line is longer than the width of the screen and the cursor is positioned at the far right position on the screen, then the screen is scrolled.

Move Cursor Up

Default key: **UP**

This command moves the cursor up one line. The horizontal position of the cursor does not change. If the line above the current line is shorter than the current line, then the cursor is positioned at the end of the line above the current line. If the cursor is currently positioned at the first line on the screen, then the screen is scrolled.

Move Cursor Down

Default key: **DOWN**

This command moves the cursor down one line. The horizontal position of the cursor does not change. If the line below the current line is shorter than the current line, then the cursor is positioned at the end of the line below the current line. If the cursor is currently positioned at the last line on the screen, then the screen is scrolled.

Move Cursor One Word Left

Default key: **Ctrl-LEFT**

This command moves the cursor to the beginning of the previous word, even if the previous word is on a previous line. If necessary, the screen is scrolled.

Move Cursor One Word Right

Default key: **Ctrl-RIGHT**

This command moves the cursor to the beginning of the next word, even if the next word is on a subsequent line. If necessary, the screen is scrolled.

Move Cursor to Line Start

Default key: **F5**

This command moves the cursor to the beginning of the current line. If necessary, the screen is scrolled.

Move Cursor to Line End

Default key: **F6**

This command moves the cursor to the end of the current line. If necessary, the screen is scrolled.

Move Cursor to First Line of Buffer

Default key: **HOME**

This command moves the cursor to the first character of the first line of the current buffer.

Move Cursor to Last Line of Buffer

Default key: **END**

This command moves the cursor to the first character of the last line of the current buffer.

Move Cursor to First Line of Window

Default key: **Ctrl-HOME**

This command moves the cursor to the first character of the first line on the current screen.

Move Cursor to Last Line of Window

Default key: **Ctrl-END**

This command moves the cursor to the first position of the last line of the current screen.

Page Up

Default key: **PGUP**

This command displays the previous screen of the file.

Page Down

Default key: **PGDN**

This command displays the next screen of the file.

Scroll Window Up

Default key: **Ctrl-UP**

This command scrolls the current window up one line without changing the relative cursor position on the screen.

Scroll Window Down

Default key: **Ctrl-DOWN**

This command scrolls the current window down one line without changing the relative cursor position on thescreen.

Go to Line Number

Default key: **Ctrl-G**

This command causes the message **Line number:** to appear on the response line. If a valid line number is entered, the cursor is placed on the first character of the specified line. If the line number entered is less than one or greater than the total number of lines in the file, an error message is displayed, and the user is prompted for another line number. If only the enter key is pressed in response to the prompt, the cursor is placed on the first line of the marked block of lines, regardless of which window is currently active.

Change Insert/Over Strike Mode

Default key: **INSERT**

This command is used to toggle between insert mode and over strike mode.

In insert mode, a character typed at the keyboard is inserted at the current cursor position. All the characters on the current line that are to the right of the current cursor position are moved to the right one position, and the length of the line is increased by one.

In over strike mode, every character typed at the keyboard over writes the character that is at the current cursor position. The characters to the right of the current cursor position are unchanged, and the length of the line does not change. In over strike mode, an **O** is displayed in reverse video in the upper right corner of the screen.

New Line Before

Default key: **Ctrl-B**

This command inserts a new line directly above the current line. The cursor is moved to the first position of the newly inserted line.

New Line After

Default key: **Ctrl-A**

This command inserts a new line directly after the current line. The cursor is moved to the first position of the newly inserted line.

Split Current Line

Default key: **ENTER**, **RETURN**, or **Ctrl-M**

This command splits the current line at the current cursor position. All the characters to the right of the current cursor position are moved to the next line. The cursor is moved to the first position of the newly created line.

Backspace

Default key: **Ctrl-H** or **BACKSPACE**

This command deletes the character immediately preceding the character at the current cursor position. All the characters to the right of the deleted character on the current line are moved to the left one position. The line length is decreased by one. If the cursor is currently at the first position of a line, no change occurs.

Delete Character

Default key: **DELETE**

This command deletes the character at the current cursor position. All the characters to the right of the deleted

character on the current line are moved to the left one position. The line length is decreased by one. If the cursor

is positioned directly after the last character on a line, then the line following the current line is appended to

the current line.

Delete Line

Default key: **Ctrl-D**

This command deletes the current line. The lines following the current line are moved up by one line. The cursor is positioned at the first character on the line that replaces the deleted line.

Delete to Line End

Default key: **Ctrl-L**

This command deletes all the characters on the current line to the right of the cursor, including the character at the current cursor position. The cursor position remains unchanged. If the cursor is positioned beyond the last character on a line, no change occurs.

Undo Deleted Line

Default key: **Ctrl-U**

This command is used to re-insert the most recently deleted line. The line is inserted directly above the line containing the cursor. Up to 100 previously deleted lines can be recovered using the Undo Deleted Line command.

Mark Block

Default key: **ESC**

This command is used to create a block or group of lines that can be copied, moved, deleted, or written to a new file. This command is issued twice to mark a block of lines: once on the first line of the block of lines and again on the last line. The order in which the lines are marked is not important. All lines between and including the first and the last line constitute the block. The block of lines is displayed in reverse video. For all current windows, only one block at a time can be marked. If the buffer containing the marked block is swapped for another, the marked block is automatically unmarked.

The escape key may be pressed a third time to unmark a block. When a block is unmarked, the lines are no longer displayed in reverse video.

Copy Marked Block

Default key: **Ctrl-C**

This command is used to copy a block of lines to a new place in a file. The block is inserted directly after the line containing the cursor. The original marked block of lines remains unchanged, so the file contains two identical copies of the group of lines. Only the original marked block is displayed in reverse video.

Move Marked Block

Default key: **Ctrl-N**

This command is used to move a block of lines to a new place in a file. The block is inserted directly after the line containing the cursor, and the block is still displayed in reverse video.

Delete Marked Block

Default key: **Ctrl-K**

This command deletes a marked block of lines.

Search Forward

Default key: **Ctrl-S**

This command is used to search for a string of characters. The message **Search Pattern:** is displayed on the response line. The search string entered by the user can be up to 30 characters long. Only print able characters and blanks are allowed in the search string.

By default, the **$** character as the first character of the search string is used to search for a string of characters that is anchored at the start of a line. The **!** character as the last character of the search string is used to search for a string of characters that is anchored at the end of a line. The **?** character is used as the wild card character. The **#** forcing character is used to search for the wild card character. These default characters can be altered on the option settings screen.

If the up arrow key is pressed in response to this prompt, the last search pattern entered is retrieved.

The search begins at the current cursor position. The entire file is searched, including lines that are longer than the width of the screen. If a match is found, the cursor is positioned at the beginning of the string and the message **Found** is displayed on the response line. Otherwise, the message **Not found** is displayed on the response line.

Searching may be case sensitive. Check the **Case-Sensitive Search** option on the option settings screen.

Search Again

Default key: **Ctrl-Z**

This command repeats the last **Search Forward**.

Translate

Default key: **Ctrl-T**

This command translates a string of characters to a new string of characters. The message **Translate from:** is displayed on the response line. The search pattern can be up to 30 characters long. The search pattern may contain the **?**, **#**, **$** and **!** characters as described in the Search Forward section.

After a search pattern is entered, the message **Translate to:** is displayed. The replacement pattern can be up to 30 characters long. The replacement pattern may contain the **?** and **#** characters as described in the Search Forward section.

The replacement pattern may also contain the **^** character. By default, this character is used to insert the most recently matched search string. For example, assume the last search pattern was **a?c** and the string **abc** was found. If a replacement pattern of **^d** is specified, the replacement string becomes **abcd**. The default insertion character, **^**, can be altered on the option settings screen.

If the up arrow key is pressed in response to these prompts, the last translate values entered are retrieved.

The search begins at the current cursor position. If a match is found, the cursor is positioned at the beginning of the string, and the message, **local(L), global(G), marked (M), final(F), skip(space bar), or quit(X)?** is displayed on the response line.

If the user enters **L**, the replacement is made, and the search continues. If another occurrence of the search string is found, the message is displayed again.

If the user enters **G**, the replacement is made, and all other occurrenc es of the search string are replaced by the replacement string.

If the user enters **M**, the replacement is made only to all occurrences of the search string found within the block of marked lines.

If the user enters **F**, the replacement is made, but the search does not continue and the Translate command is terminated.

If the user presses the space bar, the replacement is not made, but the search continues. If another occurrence of the search string is found, the message is displayed again.

If the user enters **X**, the replacement is not made and the Translate command is1 terminated.

When all translations are completed, the total number of translations made is displayed.

Searching may be case sensitive and is controlled by the **Case-Sensitive Search** option on the option settings screen. The case sensitivity option has no effect on the case of replacement characters.

Read File

Default key: **Ctrl-E**

This command reads a new file. If any changes were made to the current file before the Read File command is issued, the message, **OK to lose changes?** is displayed on the response line.

If the user types a letter **Y** or **y**, then all changes made since the last time the file was written to disk are lost. The prompt **File to edit:** is displayed on the response line. The user may enter the name of the new file to be edited. If the specified file cannot be found, the message: File not found - new file assumed is displayed.

If the user responds with anything other than a letter y to the OK to lose changes? message, then no action takes place.

Write File

Default key: **Ctrl-W**

This command writes the current file to disk. If a copy of the file already exists on disk, then the name of the older copy is changed so that it has three underscore characters (_ _ _) as the extension. In this way, the newly edited file is always written to disk with the orginal name and the user always has a copy of the file as it was before the most recent editing session.

Write Marked Block

Default key: **Ctrl-V**

This command causes the marked block to be written to disk. The message **File to Write:** is displayed on the response line. This command can only be used when a block is currently marked using the Mark Block command.

Change File Name

Default key: **Ctrl-F**

This command is used to change the name of the file being edited. The prompt **New file name:** is displayed on the response line. This command does not write the file to disk. It simply changes the name of the file in memory. The file is not written to disk until the Write File command is issued.

Merge File

Default key: **Ctrl-Y**

This command copies the lines contained in one file to the file currently being edited. When this command is invoked, the message **File to merge:** is displayed on the response line. A copy of the

lines of the new file are inserted directly above the line containing the cursor. Changes to these lines do not affect the merged file in any way. The name of the current file does not change.

Change Edit Window

Default key: **F1**

This key is used to edit text in another window. A maximum of two windows may be open simultaneously.

The full-screen window is Window 1. If the **F1** key is pressed when only Window 1 exists, then the current window is split into two smaller windows. The split happens at the line number specified in the **Window Split Line** option on the option settings screen. The screen below the horizontal line is cleared and the cursor is positioned in the upper left corner of the second window. This new window is Window 2. All the commands that can be executed in Window 1 can be executed in Window 2. The user can edit a new file, make changes, save the new file, copy text from the file in one window to the file in the other window, etc.

This key also determines which window is currently active. Control may be transferred among the windows by pressing the F1 key. The cursor moves to Window 1 when the **F1** key is pressed. The cursor moves to Window 2 when the **F1** key is pressed again.

Change to Next Buffer

Default key: **F3**

This command allows the user to edit up to nine files at one time. Each file is contained in a different buffer. This command places a specified buffer into the active window. Only a buffer contained in a window can be edited.

The buffer(s) that may be edited in a window are specified on the option settings screen. When the **F3** key is pressed, the next buffer is placed in the active window. For example, assume the option settings screen specifies that buffers 1, 3, and 9 may be displayed in window 1. If buffer 3 is currently displayed and the **F3** key is pressed, buffer 9 will be displayed in window 1.

Change to Previous Buffer

Default key: **F4**

This command is similar to the Change to Next Buffer command, except that it changes to the prior buffer instead of the next buffer for the active window.

Close Lower Window

Default key: **F2**

This command restores the first window (window 1) to the full size of the screen and closes the lower edit window. The text in the other window is not lost. The user can issue the Change Edit Window command, and the text in the second window will be displayed in the exact condition it was in before the Close Lower Window command was issued. The Close Lower Window command is ignored unless two windows are open.

Key Recorder On/Off

Default key: **Ctrl-R**

This command turns on and off the key recorder feature. When this command is issued, a letter **R** is displayed in reverse video in the upper right corner of the screen. Any keys pressed after that point are recorded. When this command is issued a second time, key recording is turned off. This command is used in conjunction with the Playback Recorded Keystrokes command.

Playback Recorded Keystrokes

Default key: **Ctrl-P**

This command is used in conjunction with the Key Recorder command. It causes the previously recorded keys to be played back. This is a very useful feature because it allows the user to enter a complicated series of commands and key strokes, and then repeat those commands any number of times with out having to retype.

Save Recorded Keystrokes in File

Default key: F7

This command allows the user to save recorded keystrokes in a file. When this command is issued, the message **File to write re corded keystrokes to:** is displayed. Enter the file name. The recorded key strokes will be saved in the specified file. A default extension of **.rec** will automatically be appended to the file name, replacing the extension if it was specified.

Playback Recorded Keystrokes in File

Default key: **F8**

This command allows the user to play back any recorded key strokes that were previously saved in a file using the Save Recorded Keystrokes in File command. When this command is issued, the message **Playback file name:** is displayed. Enter the file name. The contents of the specified file will be played back.

Nested files may not be played back. For example, assume key strokes were saved in a file called **nestfile.rec**. A user presses **Ctrl-R** to begin recording and types the letters **abc**. Then the user uses the **F8** key to play back the contents of **nestfile.rec**. He then presses **Ctrl-R** to stop recording and uses **F7** to save the key strokes to **newfile.rec**. When the user plays back **newfile.rec** using **F8**, only the letters **abc** are displayed. The contents of the nested keystroke file are ignored.

Playback File Keystrokes Again

Default key: **Ctrl-Q**

This command plays back the contents of the file that were last played back using F8.

Set Options

Default key: **Ctrl-O**

This command is used to view or set options and keystrokes that execute the edit commands. When the Set Options command is issued, the option settings screen containing all the edit commands is displayed. The arrow (**->**) points to the current option. To change the value of that option, press the **C** key (change). Changes are effective immediately. To scroll the pointer to the current option, press the **U** key (up) or the **D** key (down). Press the **P** key (page) to change to the next option page. Press the **X** key (exit) to return normal edit mode.

encode

The encode utility allows files of any format to be converted to standard 80 characters per line text files and then back to binary files. This procedure is typically used to allow transmission of **.dbc** type files via mechanisms that do not allow binary mode transmission. The format of the encode command line is:

encode *file1* [*file2*] [**-cfg=***filename*] [**-d**] [**-j**[**r**]] [**-o=***filename*]

file1 is the name of the input file. If no extension is specified and the **-d** parameter is not specified, **.dbc** is assumed. If no extension is specified and the **-d** parameter is specified, **.txt** is assumed. The search path for the input file is controlled with the **dbcdx.file.open** runtime property.

file2 is the name of the output file. If *file2* is not specified and **-d** is not specified, *file1* with an extension of **.txt** is assumed. If *file2* is not specified and **-d** is specified, *file1* with an extension of **.dbc** is assumed. If *file2* is specified without an extension and **-d** is not specified, **.txt** is assumed. If *file2* is specified without an extension and **-d** is specified, **.dbc** is assumed. The search path for the input file is controlled with the **dbcdx.file.open** and **dbcdx.file.prep** runtime properties.

-cfg=*filename* is the runtime property file.

-d is the decode parameter. If not specified, the input binary file is encoded into the output text file. If **-d** is specified, the input text file is decoded into the output binary file.

-j[**r**] is the share open mode parameter. The **-j** parameter causes the input file to be opened in share read/write mode. The **-jr** parameter causes the input file to be opened in share read-only mode.

-o=*filename* is the options file parameter. *filename* is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line. when the number of options is too large to fit on one command line.

The encode utility returns a return code of zero if it completes successfully. Otherwise, it returns a non-zero return code. If the return code is non-zero and encode is running under chain execution, the chain execution is aborted.

exist

The exist utility tests for the existence of a file. The exist utility can detect a file that is opened exclusively by a task that is executing simultaneously. The format of exist the command line is:

`exist` *file1* [**`-a`**] [**`-c`**] [**`-cfg=`***filename*] [**`-d`**] [**`-f`**] [**`-l`**] [**`-p`**] [**`-s`**] [**`-w`**]

file1 is the name of the file. If no extension is specified, **.txt** is assumed. The search path for the file is controlled by the **dbcdx.file.open** runtime property.

`-a` is the search all file paths parameter. It is has the same effect as specifying all other parameters in dividually.

`-c` is the search using the **dbcdx.file.editcfg** runtime property.

`-cfg=`*filename* is the runtime property file.

`-d` is the search using the **dbcdx.file.dbc** runtime property.

`-f` is the search using the **dbcdx.file.open** runtime property.

`-i` is the search using the **dbcdx.file.image** runtime property.

`-p` is the search using the **dbcdx.file.prep** runtime property.

`-s` is the search using the **dbcdx.file.source** runtime property.

`-w` is the search current directory parameter.

If the file is found, the exist utility displays the file name and the file path, and returns a return code of zero. If the file is not found, the exist utility displays the message **File does not exist**, and returns a non-zero return code. If the return code is non-zero and exist is running under chain execution, the chain execution is aborted.

filechk

The filechk utility checks the content of files. It is used to diagnose corrupted files or other problems. The format of the filechk command line is:

filechk *file1* [**-a**] [**-c**] [**-cfg=***filename*] [**-e**] [**-f**] [**-i**] [**-j**[**r**]] [**-l=***n*] [**-o=***filename*] [**-p**] [**-***t=type*] [**-x**]

file1 is the name of the input file. If no extension is specified, **.txt** is assumed. The search path for the input file is controlled with the **dbcdx.file.open** runtime property.

-a is the AIM file information parameter. Information about the **.aim** file is displayed.

-c is the change parameter. If **-c** is specified, when any statistic is changed, all of the statistics are displayed. If **-c** is not specified, the statistics are only displayed when the end of file is found.

-cfg=*filename* is the runtime property file.

-e is the turn off exit status parameter. This parameter causes the exit code not to be set to one when an eof character is en countered in the middle of the file. This is useful to prevent a chain from aborting.

-f is the fix end-of-file parameter. This parameter causes filechk to modify the end of file to correct the eof character. The **-t** parameter can be used to specify the type of file to modify. Otherwise, filechk tries to determine the file type. If the file type cannot be determined, then filechk displays a message and does nothing.

-i is the INDEX file information parameter. Information about the **.isi** file is displayed.

-j[**r**] is the share open mode parameter. The **-j** parameter causes the input file to be opened in share read/write mode. The **-jr** parameter causes the input file to be opened in share read-only mode.

-l=*n* is the record length parameter. *n* is a number between 1 and 65500. filechk displays a message whenever it encounters a record with length different than *n*.

-o=*filename* is the options file parameter. *filename* is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

-p is the program file information parameter. This parameter displays information about the **.dbc** file created by dbcmp.

-t=*type* is the force file type option. **type** can be **data**, **crlf**, **text**, or **std**. filechk determines the type of file from the last 1 or 2 bytes of the file. If the last byte(s) are invalid, this option is helpful to prevent filechk from aborting while determining the text file type. It is suggested that this option also be specified when using the **-f** option. Under Windows, **text** is the equivalent to **crlf**; otherwise, **text** is the equivalent to **data**.

-x is the extra information parameter. For **.txt** files, this parameter causes a special message to be displayed for each record that contains a non-printable control character. For **.aim** and **.isi** files, other information is printed.

index

The index utility creates a special index file from a text file. The special index file is used by DB/C DX for IFILE type files. The format of the index command line is:

index *file1* [*file2*] *key-spec* [*key-spec...*] [**-a=n**] [**-b=***n*] [**-cfg=***filename*] [**-d**] [**-e**] [**-f=***filename*] [**-j**[**r**]] [**-k=***keytagfile*] [**-o=***filename*] [**or**] [**-p***n*[**-***n*]**eq***c*[*c...*]] [**-p***n*[**-***n*]**ne***c*[*c...*]]] [**-p***n*[**-***n*]**gt***c*[*c...*]]] [**-p***n*[**-***n*]**ge***c*[*c...*]]] [**-p***n*[**-***n*]**lt***c*[*c...*]]] [**-p***n*[**-***n*]**le***c*[*c...*]] [**-r**] [**-s**[**=***n*]] [**-t**] [**-w=***filename*] [**-x**] [**-y**]

file1 is the name of the input text file. If no extension is specified, **.txt** is assumed. This file can be any text type file. The search path for the input file is controlled with the dbcdx.file.open runtime property.

file2 is the name of the output index file. If *file2* is not specified, *file1* with an extension of **.isi** is assumed. If *file2* is specified without an extension, **.isi** is assumed. The search and create path for the output file is controlled with the **dbcdx.file.open** and dbcdx.file.prep runtime properties.

key-spec is a key specification. Valid key specifications are of the form *nnn* or *nnn*-*nnn* where *nnn* is the character position and *nnn*-*nnn* is the range of character positions from each input file text record that is used as a key. *nnn* is a number between 1 and 65500. More than one key specification may be designated. The maximum length of all keys is 255 characters.

-a=*n* is the memory allocation parameter where *n* is a one to three-digit number. n multiplied by 1024 is the number of bytes of memory that is allocated for this utility. This parameter affects only the performance of the index command.

-b=*n* is the block size parameter. Valid block size values are limited to 512, 1024, 2048, 4096, 8192, and 16384. The default block size value is 1024. In some cases, increasing this parameter can increase performance.

-**cfg=***filename* is the runtime property file.

-d is the duplicate keys allowed parameter. If not specified, duplicate keys are not allowed in the file and an error message is displayed if any duplicates are encountered. **-d** and **-f** are mutually exclusive.

-e is the existing command line parameters option. When this option is specified, the index command utilizes the command line parameters that are stored in the existing **.isi** file. If the existing **.isi** file was created using the prepare statement rather than the index command, the command line parameters will not be present in the **.isi** file and the **-e** option cannot be used. **-e** is mutually exclusive with all other command line options.

-f=*filename* is the duplicate key file option. When this option is specified, if any duplicate keys are encountered, they are written to the DB/C type file named filename. **-d** and *-f* are mutually exclusive.

-j[**r**] is the share open mode parameter. The **-j** parameter causes the input file to be opened in share read/write mode. The **-jr** parameter causes the input file to be opened in share read-only mode.

-k=*keytagfile* is the keytag input file option. When this option is specified, the keytag file contains records that contain keys and record positions. In the keytag file, the first 12 characters of each record are the text file position and the rest of the record is the key. All records in the keytag file must be the same length. *file1* is not read, but its filename is stored in the output file, *file2*.

-o=*filename* is the options file parameter. *filename* is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

or is the logical or parameter. If the **or** parameter is placed between two **-p** parameters, they are logically OR-ed together instead of logically AND-ed together. The precedence of all **-p** operations is from left to right.

-p*n*[**-***n*]**eq***c*[*c*...]
-p*n*[**-***n*]**ne***c*[*c*...]
-p*n*[**-***n*]**gt***c*[*c*...]
-p*n*[**-***n*]**ge***c*[*c*...]
-p*n*[**-***n*]**lt***c*[*c*...]
-p*n*[**-***n*]**le***c*[*c*...] are the record select range parameters. *n* may be a one to four digit decimal number. *c* is the match character. The key of a record is included in the index if the record contains a character that is equal, not equal, greater than, greater than or equal, less than, or less than or equal to (respectively) the match character at character position *n*. Otherwise, the record is ignored. **=** may be used instead of **eq**. **#** may be used instead of **ne**. The optional *n*-*n* is used to specify a range of character positions. If a range is specified, the number of match characters specified should be the same as the number of positions in the range. If the number of match characters specified is less than the number of positions in the range, it is assumed that the unspecified match characters are blanks. If the number of match characters specified is greater than the number of positions in the range, the extra match characters are truncated. If more than one **-p** parameter is specified, they are logically AND-ed together. For example:

-p5eqxy

causes the records with a match character at the fifth character position equal to **x** or equal to **y** to be selected.

-r is the rename file parameter. This parameter changes the name of the text file stored in the index file to the specified input text file name. This parameter is mutually exclusive with all other parameters. Both the input text file and the output index file must be specified.

-s[**=***n*] is the space reclamation parameter. If this parameter is specified, records written to the file will reuse the space made available by deleted records whenever possible. This parameter can only be specified if the records in the file are decompressed and all the same length. The actual record size (*n*) only needs to be specified if the input file is initially empty. *n* is a number between 1 and 65500. Read performance is not affected by this parameter. Write performance is usually slightly worsened if this parameter is specified.

-t is the limited text file search parameter. If this parameter is specified, the utility will only look for the text file in the same directory in which the index file is found.

-w=*filename* is the work file parameter. If not specified, the default work file name is **sort.wrk**.

-x is the reverse selection parameter. This parameter causes those records that are selected by the **-p** parameter to be excluded from the output file and those records not selected to be written into the output file.

-y is the test end of file parameter. This parameter causes aimdex to fail if an end of file character is encountered other than at the physical end of file.

The index utility creates a work file in the current directory. The amount of space in bytes that must be available for this file is:

1.6 * *k* * *n* where *k* is the key length plus 6, and *n* is the number of records to be indexed

If the **dbcdx.file.collate** runtime property is specified, it is used to translate each record and determine the record's position in the index.

The index utility returns a return code of zero if index completes successfully. The return code is non-zero if index does not complete successfully. If the return code is non-zero and index is running under chain execution, then chain execution is aborted.

library

The library utility provides access to the library manager utility. The format of the command line is:

library *file1* [**-a=***filename* [*membername*]] [**-c**] [**-cfg=***filename*] [**-d=***membername*] [**-e=***name* [*filename*]] [**-j**[**r**]] [**-l**] [**-n=***membername1 membername2*] [**-o=***filename*] [**-r=***filename* [*membername*]] [**-v**]

file1 is the name of the library. If the extension is not specified, **.lib** is assumed. If *file1* does not exist, it is created. The search path is the current directory only.

-a=*filename* [*membername*] is the add parameter. *filename* is the name of the file to be added to the library. If an extension is not specified, **.txt** is assumed. *membername* is the optional member name. If not specified, the member name is the same as *filename*.

-c is the compression parameter. A new library is created and all the members of the original library are copied to the new library in compressed format. Then the original library is deleted and the new library is renamed to the name of the original library.

-cfg=*filename* is the runtime property file.

-d=*membername* is the delete parameter. The member specified by *membername* is deleted from the library.

-e=*name* [*filename*] is the extract parameter. The member specified by name is extracted from the library. If the optional *filename* exists, it is extracted to the name filename. If no extension is specified, **.txt** is assumed. If the optional filename does not exist, then it is extracted to a file with filename the same as the member name with a **.txt** extension.

-j[**r**] is the share open mode parameter. The **-j** parameter causes the input file to be opened in share read/write mode. The **-jr** parameter causes the input file to be opened in share read-only mode.

-l is the list parameter. The names of all the library members, sizes, and time stamps are displayed.

-n=*membername1 membername2* is the rename parameter. The member called *membername1* is renamed to *membername2*.

-o=*filename* is the options file parameter. *filename* is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

-r=*filename* [*membername*] is the replace parameter. filename is the name of the file to be replaced in the library. If an extension is not specified, **.txt** is assumed. *membername* is the optional member name. If not specified, the member name is the same as filename.

-v is the ignore **dbcdx.display** and **dbcdx.keyin** runtime properties.

If the library utility is invoked with any command line parameter except **-o** or **-j**, then the specified operation takes place. Parameters can be specified multiple times. They are executed in the order in which they appear on the command line.

If the library utility is invoked with no command line parameters or with the **-o** or **-j** parameters, then the library command executes in interactive mode. The **cmd:** prompt is displayed.

Valid commands are as follows:

a=*filename* [*membername*]	add file to library
r=*filename* [*membername*]	replace file in library
e[**=***membername*]	filename extract file from library
d=*membername*	delete file from library
n=*membername membername*	rename file in library
l	list the members, sizes, and time stamps
h	display help screen
q	quit

A comma may be specified between names. These commands work the same way as the corresponding parameters on the command line. Only one command can be executed at a time. After a command completes its task, the **cmd:** prompt is displayed again.

list

The list utility displays a text file on the screen. The format of the list command line is:

list *file1* [**-a=***key*] [**-b=***n*] [**-c=***n*] [**-cfg=***filename*] [**-e=***n*] [**-f**] [**-h=***string*] [**-i**[**=***key*]] [**-j**[**r**]] [**-l=***n*] [**-n**[**=***n*]] [**-o=***filename*] [**or**] [**-p***n*[**-***n*]**eq***c*[*c*...]] [**-p***n*[**-***n*]**ne***c*[*c*...]]] [**-p***n*[**-***n*]**gt***c*[*c*...]]] [**-p***n*[**-***n*]**ge***c*[*c*...]]] [**-p***n*[**-***n*]**lt***c*[*c*...]]] [**-p***n*[**-***n*]**le***c*[*c*...]] [**-r=***n*] [**-s=***n*] [**-t=***n*] [**-x**[**=***n*]]

file1 is the name of the file to be listed. If no extension is specified, .txt is assumed. This file can be any text type file. The search path for the input file is controlled with the **dbcdx.file.open** runtime property.

-a=*key* is the aimdex parameter. If specified, the records associated with key are listed. If the name of the file does not have an extension, **.aim** is assumed. The **-a**, **-i**, **-r**, and **-s** parameters are mutually exclusive.

-b=*n* is the bottom margin parameter. *n* is a one to three-digit number that is the number of blank lines at the bottom of each page of the output. The **-b** and the **-f** parameters are mutually exclusive.

-c=*n* is the column parameter. *n* is a one to four-digit number between 1 and 65500. When this parameter is specified, the records are displayed starting with column number *n* in each record. If not specified, the default starting column is one.

-cfg=*filename* is the runtime property file.

-e=*n* is the list last record parameter. The **-e** parameter lists the last *n* records that would be displayed. *n* is a number between 1 and 999, inclusive. The **-e** parameter works for both sequential and indexed access.

-f is the print formatted file parameter. The file to be listed is assumed to contain standard ASCII carriage control characters in the first position of each record. The **-n** parameter and the **-f** parameter are mutually exclusive.

-h=*string* is the header parameter. This parameter causes the string of characters and the page number to appear on the third line of each page of the output. If the **-h** parameter is specified and the **-p** parameter is not specified, the number of lines on each page is defaulted to 57. If the **-t** parameter is not specified, the top margin is six lines long. If the **-b** parameter is not specified, the bottom margin is three lines long. The **-h** and **-f** parameters are mutually exclusive.

-i[**=***key*] is the index parameter. If the **-i** parameter is specified, the records are listed in key sequential order. If key is also specified, then the records are listed in key sequential order starting with the record associated with key. If the name of the file does not have an extension, **.isi** is assumed. The **-a**, **-i**, **-r**, and **-s** parameters are mutually exclusive.

-j[**r**] is the share open mode parameter. The **-j** parameter causes the input file to be opened in share read/write mode. The **-jr** parameter causes the input file to be opened in share read-only mode.

-l=*n* is the line size parameter, where *n* is the maximum line width in characters. *n* is a number between 1 and 256. If a line is longer than *n* characters, then only *n* characters of the line are listed. If this parameter is not specified and **-f** is not specified, the default line width is 79. If this parameter is not specified and **-f** is specified, the default line width is 256.

-n[**=***n*] is the numbering parameter. If specified, line numbers are displayed to the left of each line as it is displayed. If *n* is specified, then *n* specifies the width in characters of the line numbers. The default width is 6. The **-n** parameter and the **-f** parameter are mutually exclusive.

-o=*filename* is the options file parameter. *filename* is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

or is the logical or parameter. If the **or** parameter is placed between two **-p** parameters, they are logically OR-ed together instead of logically AND-ed together. The precedence of all **-p** operations is from left to right.

-p*n*[**-***n*]**eq***c*[*c*...]
-p*n*[**-***n*]**ne***c*[*c*...]
-p*n*[**-***n*]**gt***c*[*c*...]
-p*n*[**-***n*]**ge***c*[*c*...]
-p*n*[**-***n*]**lt***c*[*c*...]
-p*n*[**-***n*]**le***c*[*c*...] are the record select range parameters. *n* may be a one to four digit decimal number. *c* is the match character. The record is listed if the record contains a character that is equal, not equal, greater than, greater than or equal, less than, or less than or equal to (respectively) the match character at character position *n*. Otherwise, the record is ignored. **=** may be used instead of **eq**. **#** may be used instead of **ne**. The optional *n*-*n* is used to specify a range of character positions. If a range is specified, the number of match characters specified should be the same as the number of positions in the range. If the number of match characters specified is less than the number of positions in the range, it is assumed that the unspecified match characters are blanks. If the number of match characters specified is greater than the number of positions in the range, the extra match characters are truncated. If more than one **-p** parameter is specified, they are logically AND-ed together. For example:

-p5eqxy

causes the records with a match character at the fifth character position equal to **x** or equal to **y** to be selected.

-r=*n* is the record number parameter. *n* is a one to nine-digit record number. This parameter causes the file to be listed starting at the record number specified. The **-a**, **-i**, **-r**, and **-s** parameters are mutually exclusive.

-s=*n* is the starting position parameter. *n* is a one to nine-digit number that is the file position in bytes. This parameter causes the file to be listed starting at the specified starting position. The **-a**, **-i**, **-r**, and **-s** parameters are mutually exclusive.

-t=*n* is the top margin parameter. *n* is a one to three-digit number that is the number of blank lines at the top of each page of the output. If the **-t** parameter is specified and the **-h** parameter is not specified, then the file name (*file1*) and the page number appear on the third line of each page of the output. If **-h** is specified, then the string appears on the third line of each page of the output. In either case, if *n* is less than three, then no header line appears. The **-t** and **-f** parameters are mutually exclusive.

-x[**=***n*] without =*n* is the reverse selection parameter. This parameter causes those records that are selected by the **-p** parameter to be excluded from the output file and those records not selected to be written into the output file. **-x** with =*n* specified is the tab position specifier parameter. *n* is a number between 1 and 65500 that is the position of the character that the tab character represents. This parameter may be specified up to 30 times.

The list utility returns a return code of zero if list completes successfully. The return code is non-zero if list does not complete successfully. If the return code is non-zero and list is running under chain execution, then chain execution is aborted.

reformat

The reformat utility copies an input text file to an output text file and re for mats the new file. Deleted records are not copied. The syntax of the reformat command line is:

reformat *file1 file2* [*field-spec...*] [**-a**] [**-b=***nnn*] [**-c**] [**-cfg=***filename*] [**-f=***string*] [**-i**]
[**-fp***n*[**-***n*]**eq***c*[*c...*] *r*[*r...*]] [**-fp***n*[**-***n*]**ne***c*[*c...*] *r*[*r...*]] [**-fp***n*[**-***n*]**gt***c*[*c...*] *r*[*r...*]]
[**-fp***n*[**-***n*]**ge***c*[*c...*] *r*[*r...*]] [**-fp***n*[**-***n*]**lt***c*[*c...*] *r*[*r...*]] [**-fp***n*[**-***n*]**le***c*[*c...*] *r*[*r...*]]
[**-fr***n*[**-***n*]**=***filename*] [**-frb***n*[**-***n*]**=***filename*] [**-j**[**r**]] [**-k**] [**-l=***n*] [**-n=***filename*]
[**-o=***filename*] [**or**] [**-p***n*[**-***n*]**eq***c*[c...]] [**-p***n*[**-***n*]**ne***c*[*c...*]] [**-p***n*[**-***n*]**gt***c*[*c...*]]
[**-p***n*[**-***n*]**ge***c*[*c...*]] [**-p***n*[**-***n*]**lt***c*[*c...*]] [**-p***n*[**-***n*]**le***c*[*c...*]] [**-r**] [**-s=***filename*]
[**-t**[**=***type*]] [**-v**] [**-x**[**=***n*]] [**-y**] [**-z=***nnn*] [**-zc=***nn*] [**-zd=***nnn*] [**-zr=***nnn*] [**-zx=***nnn*]

file1 is the name of the input file. If no extension is specified, **.txt** is assumed. This file can be any text type file. The name of the input file and the name of the output file must be different. The search path for the input file is con trolled with the **dbcdx.file.open** runtime property.

file2 is the name of the output file. If no extension is specified, **.txt** is assumed. The name of the input file and the name of the output file must be different. The search and create path for the output file is controlled with the **dbcdx.file.open** and **dbcdx.file.prep** runtime properties.

field-spec is the field specification parameter. It may be included multiple times. If this parameter is not specified, the entire record is copied from the input file to the output file. Valid field specifications are of the form *nnnn* or *nnnn*-*nnnn*, where *nnnn* is a character position and *nnnn*-*nnnn* is a range of character positions. *nnnn* is a one to four-digit decimal number from 1 to 65500. The contents of the input records at the character positions are appended to the specified output record.

-a appends the records from the input file to the output file. This implies that the output file must already exist. If the **-a** parameter is not specified, then the records are placed into the output file as the first records, overwriting any records that already exist. If this parameter is specified, then the output file type is not changed, and the **-t** parameter is ignored if it is selected.

-b=*nnnnn* is the field blank parameter. This parameter is used to insert *n* blanks into the reformatted records. It is typically used in conjunction with the field specification parameter and may be included multiple times.

-c means that the output file is of DB/C-type compressed format. The **-c** and **-t** parameters are mutually exclusive.

-cfg=*filename* is the runtime property file.

-f=*string* is the field string parameter where string is a string of characters enclosed in quotation marks. This parameter is used to insert a string of characters into the reformatted records. It is typically used in conjunction with the field specification parameter and may be included multiple times. If double quotation marks must be contained in the character string, the quotation marks must be preceded with a back slash character (****).

[**-fp***n*[**-***n*]**eq***c*[*c...*] *r*[*r...*]]
[**-fp***n*[**-***n*]**ne***c*[*c...*] *r*[*r...*]]
[**-fp***n*[**-***n*]**gt***c*[*c...*] *r*[*r...*]]
[**-fp***n*[**-***n*]**ge***c*[*c...*] *r*[*r...*]]
[**-fp***n*[**-***n*]**lt***c*[*c...*] *r*[*r...*]]
[**-fp***n*[**-***n*]**le***c*[*c...*] *r*[*r...*]] are the conditional string insertion parameters. *n* may be a one to four digit decimal number. *c* is the match character. *r* is the insertion character. *rr...* are the insertion characters. The insertion character or characters are inserted into only those records whose character at the nth position of the that is equal, not equal, greater than, greater than or equal, liss than, or less than or equal (respectively) the match character. **=** may be used instead of **eq**. **#** may be used instead of **ne**.

The optional *n*-*n* is used to specify a range of character positions. If a range is specified, the number of match characters specified must be the same as the number of positions in the range.

-fr*n*[**-***n*]**=***filename*
-frb*n*[**-***n*]**=***filename* are the field replacement parameters, where **fr** stands for field replacemtnt, and **frb** stands for field replacement or blank fill. *n* is a one to five digit decimal number from 1 to 65500 specifying a character position. The optional *n*-*n* specifies a range of character positions. *filename* is the name of a translate file. The translate file is a text file with records all the same length and the length must be at least one larger than the size of the field being replaced. The first *m* characters in each record, where *m* is the number of characters in the input field, is the match value. The remaining characters in each record are the replacement characters. If a match occurs, the replacement characters replace the matched characters. If no match occurs, the field is left unchanged. If **frb** is specified, the same thing happens except if no match occurs, the output field is filled with blanks. If no match occurs and the length of the replacement characters is different from the match characters length, then the extra characters are removed, or blanks are inserted, so that the resulting record length is the same regardless of whether a replacement occurred or not.

-i is the tab insert parameter. If **-i** is not specified and one or more **-x=***n* parameters are specified, the operation is tab expansion. That is, tabs are removed from the input file and expanded into blanks in the output file. If **-i** is specified and one or more **-x**=*n* parameters are specified, the operation is tab compression. In this case, the output file must be a local text file. Tabs are inserted, where appropriate, into the output file. These parameters are useful when dealing with files created and used by different text editors.

-j[**r**] is the share open mode parameter. The **-j** parameter causes the input file to be opened in share read/write mode. The **-jr** parameter causes the input file to be opened in share read-only mode.

-k is the key tag parameter. This parameter causes the output file to contain the input record file position as the first 12 characters of the output record.

-l=*n* is the length parameter. *n* is the length of the output records. n is a number between 1 and 65500. If the length of the input record is shorter than *n*, it is padded with blanks. If the length of the input record is longer than n, it is truncated.

-n=*filename* is the read translate table. filename contains the 256 byte translate table applied to each record as it is read.

-o=*filename* is the options file parameter. filename is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

or is the logical or parameter. If the **or** parameter is placed between two **-p** parameters, they are logically OR-ed together instead of logically AND-ed together. The precedence of all **-p** operations is from left to right.

-p*n*[**-***n*]**eq***c*[*c*...]
-p*n*[**-***n*]**ne***c*[*c*...]
-p*n*[**-***n*]**gt***c*[*c*...]
-p*n*[**-***n*]**ge***c*[*c*...]
-p*n*[**-***n*]**lt***c*[*c*...]
-p*n*[**-***n*]**le***c*[*c*...] are the record select range parameters. *n* may be a one to four-digit decimal number. . *c* is the match character. Only those records with the character at the nth position of the file equal, not equal, greater than, greater than or equal, less than, or less than or equal to (respectively) the match character is copied to the output file. Otherwise, the record is ignored. = may be used instead of **eq**. **#** may be used instead of **ne**.

The optional *n*-*n* is used to specify a range of character positions. If a range is specified, the number of match characters specified should be the same as the number of positions in the range. If the number of match characters specified is less than the number of positions in the range, it is assumed that the unspecified match characters are blanks. If the number of match

characters specified is greater than the number of positions in the range, the extra match characters are truncated.

If more than one -p parameter is specified, they are logically AND-ed together.

For example:

```
-p5=xy
```

causes the records with a match character at the fifth character position equal to **x** or equal to **y** to be selected.

-r is the reclaim parameter. This parameter copies the input file to the output file. Deleted records are not copied. No other reformatting takes place. This parameter can significantly decrease the time needed to reformat a file when the deletion of records is the only reformat ting necessary. The **-r** and **-t** options are mutually exclusive and do not work together.

-s=*filename* is the unselected records output file parameter. All records that are not put into *file2* are put into filename. If no extension is specified, **.txt** is assumed.

-t[**=***type*] is the output file type option. *type* can be **data**, **crlf**, **text**, or **std**. The default is to create DB/C standard text file (**std**). Under Windows, **text** is the equivalent to **crlf**; otherwise, **text** is the equivalent to **data**. If *type* is omitted, **text** is assumed.

-x[**=***n*] without the =*n* is the reverse selection parameter. This parameter causes those records that are selected by the **-p** parameter to be excluded from the output file and those records not selected to be written into the output file. **-x** with the =*n* specified is the tab position specifier parameter. This parameter works in conjunction with the **-i** parameter to convert files that contain tabs. *n* is a number between 1 and 65500 that is the position of the character that the tab character represents. This parameter may be specified up to 30 times.

-y is the test end of file parameter. This parameter causes reformat to fail if an end of file character is encountered other than at the physical end of file.

-z=*nnn* is the convert 2 character year to 4 character year parameter. This parameter may be specified one or more times. If the 2 characters at position *nnn* are 2 digits or a blank and a digit respectively, then the numeric value is compared to the date cutoff value (**-zc**) and if greater, then "**19**" is appended to the output record; otherwise, a "**20**" is appended to the output record. Additionally, the 2 characters at position *nnn* are appended to the output record with blank to zero conversion. If the 2 characters at position *nnn* are not a valid numeric value, then 2 blank characters are appended to the output record followed by the 2 characters at position *nnn*. When using the year conversion parameters, it is expected that *field-specs* will also be specified.

-zc=*nn* is the date cutoff parameter used by the date conversion parameters. This parameter may be specified one or more times. This parameter effects all year conversion parameters that follow and therefore must be specified before the date conversion parameter(s) it applies to. Years greater than this value will be considered to be year 1900. Years less than or equal to this value will be considered to be year 2000. The default value for this parameter is "**20**".

-zd=*nnn* is the convert 2 character year to 4 character year parameter with year duplication. This parameter is identical to the **-z** parameter, except if the 2 characters specified at position *nnn* is not a valid number, then the same 2 characters are appended to the output before the 2 characters at position *nnn* are appended.

-zr=*nnn* is the convert 2 character reversed year to 4 character reversed year parameter. This parameter may be specified one or more times. This parameter is identical to the **-z** parameter, except the year is assumed to be reversed by subtracting the year from 99. The 4 character output is created by taking the input year, converting to 4 characters based on the date cutoff value and then subtracting from 9999. For example "**3**" representing the reversed year 96 causes "**8003**" to be appended to the output record.

-zx=*nnn* is the convert 2 character year to 4 character year first alternate parameter. This parameter may be specified one or more times. This parameter is identical to the **-z** parameter, except if

the 2 characters at position nnn are " **0**" or "**00**", then the behavior is different. In this situation the character at position *nnn* is appended to the output record twice followed by the 2 characters at position *nnn*.

The reformat utility returns a return code of zero if reformat completes successfully. The return code is non-zero if reformat does not complete successfully. If the return code is non-zero and reformat is running under chain execution, then chain execution is aborted.

rename

The rename utility changes the specified file name. The syntax of the rename command line is:

`rename` *file1 file2* [**`-cfg=`***filename*]

file1 is the file to be renamed. If no extension is specified, **.txt** is assumed. The search path for the input file is controlled with the **dbcdx.file.open** runtime property.

file2 is the new file name. If no extension is specified, **.txt** is assumed.

`-cfg=`*filename* is the runtime property file.

The rename utility returns a return code of zero if rename completes successfully. The return code is non-zero if rename does not complete successfully. If the return code is non-zero and rename is running under chain execution, then chain execution is aborted.

sort

The sort utility arranges records. Records from the input file are sorted and placed in the output file. The syntax of the sort command line is:

sort *file1 file2 sort-spec* [*sort-spec...*] [**-a=***n*] [**-c**] [**-cfg=***filename*] [**-f**] [**-g***n***=***c*] [**-g***n***#***c*] [**-h***n***=***c*] [**-h***n***#***c*] [**-j**[**r**]] [**-k**] [**-m=***n*] [**-o=***filename*] [**or**] [**-p***n*[**-***n*]**eq***c*[c...]] [**-p***n*[**-***n*]**ne***c*[*c...*]] [**-p***n*[**-***n*]**gt***c*[*c...*]] [**-p***n*[**-***n*]**ge***c*[*c...*]] [**-p***n*[**-***n*]**lt***c*[*c...*]] [**-p***n*[**-***n*]**le***c*[*c...*]]] [**-q**] [**-r**] [**-s**] [**-t**[**=***type*]] [**-u**] [**-w=***filename*] [**-x**] [**-y**]

file1 is the name of the input file. If no extension is specified, **.txt** is assumed. This file can be any text type file. The name of the input file and the name of the output file can be the same. The search path for the input file is con trolled with the **dbcdx.file.open** runtime property.

file2 is the name of the output file. If no extension is specified, **.txt** is assumed. The name of the input file and the name of the output file can be the same. The search and create path for the output file is controlled with the **dbcdx.file.open** and **dbcdx.file.prep** runtime properties.

sort-spec is the sort specification. Valid sort specifications are of the form *nnn*, *nnn***a**, *nnn***d**, *nnn***-***nnn*, *nnn***-***nnn***a**, or *nnn***-***nnn***d** where *nnn* is a character position and *nnn***-nnn** is a range of character positions from the input text record that are to be used as the key for the sort. *nnn* is a number between 1 and 65500. If **a** is specified, then this key is sorted in ascending order.

-a=*n* is the memory allocation parameter where *n* is a one to three-digit number. *n* multiplied by 1024 is the amount of memory that is allocated for the internal work area.

-c causes the output file to be in compressed format. The **-c** and **-t** parameters are mutually exclusive.

-cfg=*filename* is the runtime property file.

-f is the file position parameter. When this parameter is specified, each output file record contains only the twelve-byte input file record position.

-g*n***=***c* is the group record equal parameter. *n* is a number between 1 and 65500. *c* is the match character which may be any character in the standard character set. This parameter causes only those records with the character *c* at position n to be sorted. All the records that do not have the character *c* at position *n* are considered header records. They are included in the output file in the same order they appeared in the input file. The group records are sorted and written to the output file following the same header record they followed in the input file. This parameter is mutually exclusive with the **-h** parameter.

-g*n***#***c* is the group record not equal parameter. *n* is a number between 1 and 65500. *c* is the match character which may be any character in the standard character set. This parameter causes only those records that do not have the character *c* at position *n* to be sorted. All the records that do have the character *c* at position *n* are considered header records. They are included in the output file in the same order they appeared in the input file. The group records are sorted and written to the output file following the same header record they followed in the input file. This parameter is mutually exclusive with the **-h** parameter.

-h*n***=***c* is the header record equal parameter. *n* is a number between 1 and 65500. *c* is the match character which may be any character in the standard character set. This parameter causes only those records with the character *c* at position *n* to be sorted. All the records that do not have the character *c* at position *n* are considered group records. They are included in the output file following the same header record they followed in the input file. The order of the group records within one group does not change. This parameter is mutually exclusive with the **-g** parameter.

-h*n***#***c* is the header record not equal parameter. *n* is a number between 1 and 65500. *c* is the match character which may be any character in the standard character set. This parameter causes only those records that do not have the character *c* at position *n* to be sorted. All the records that have the character *c* at position *n* are considered group records. They are included in the output file following the same header record they followed in the input file. The order of the

group records within one group does not change. This parameter is mutually exclusive with the **-g** parameter.

-j[**r**] is the share open mode parameter. The **-j** parameter causes the input file to be opened in share read/write mode. The **-jr** parameter causes the input file to be opened in share read-only mode.

-k is the key tag parameter. This parameter causes each output file record to contain only the twelve character input file record position followed by the characters in the key for the sort (*sort-spec*).

-m=*n* is the maximum length parameter. *n* is the maximum length of the longest record in the input file. *n* is a number between 1 and 65500. If this parameter is not specified, the length of the first record in the input file is assumed to be the maximum length for all records in the file. If a record longer than the maximum length is encountered, an error occurs and the sort does not function.

-o=*filename* is the options file parameter. *filename* is the name of a user-created file containing options. It is useful when the number of options is too large to fit on one command line.

or is the logical or parameter. If the **or** parameter is placed between two **-p** parameters, they are logically OR-ed together instead of logically AND-ed together. The precedence of all **-p** operations is from left to right.

-p*n*[**-***n*]**eq***c*[*c*...]
-p*n*[**-***n*]**ne***c*[*c*...]
-p*n*[**-***n*]**gt***c*[*c*...]
-p*n*[**-***n*]**ge***c*[*c*...]
-p*n*[**-***n*]**lt***c*[*c*...]
-p*n*[**-***n*]**le***c*[*c*...] are the record select range parameters. *n* may be a one to four digit decimal number. *c* is the match character. The record is included in the records output if the record contains a character that is equal, not equal, greater than, greater than or equal, less than, or less than or equal to (respectively) the match character at character position *n*. Otherwise, the record is ignored. **=** may be used instead of **eq**. **#** may be used instead of **ne**. The optional *n*-*n* is used to specify a range of character positions. If a range is specified, the number of match characters specified should be the same as the number of positions in the range. If the number of match characters specified is less than the number of positions in the range, it is assumed that the unspecified match characters are blanks. If the number of match characters specified is greater than the number of positions in the range, the extra match characters are truncated. If more than one **-p** parameter is specified, they are logically AND-ed together. For example:

-p5eqxy

causes the records with a match character at the fifth character position equal to **x** or equal to **y** to be selected.

-q is the alternate sort algorithm parameter. The file will be sorted using a key tag rather than using the entire record. This parameter may be useful with large records and large files. When this parameter is used, the size of the work file created by sort is significantly reduced because only keys are written to the work file rather than entire records. Depending on the instance, this parameter can significantly increase or decrease the time required for the sort to run.

-r causes a descending sort to be done.

-s causes a stable sort to be done. If this parameter is specified, all records that have equal sort keys are guaranteed to be in the same relative order as in the input file.

-t[**=***type*] is the output file type option. *type* can be **data**, **crlf**, **text**, or **std**. The default is to create DB/C standard text file (**std**). Under Windows, **text** is the equivalent to **crlf**; otherwise, **text** is the equivalent to **data**. If *type* is omitted, *text* is assumed.

-u is the unique record parameter. This parameter causes sort to output only the first record of a group of records that have the same characters in the sort fields.

-w=*filename* is the work file parameter. If not specified, the default work file name is **sort.wrk**.

-x is the reverse selection parameter. This parameter causes those records that are selected by the **-p** parameter to be excluded from the output file and those records not selected to be sorted and written into the output file.

-y is the test end of file parameter. This parameter causes sort to fail if an end of file character is encountered other than at the physical end of file.

The sort utility creates a work file in the current directory. The amount of space in bytes that must be available for this file is:

1.6 * *m* * n

where *m* is the maximum record length plus 2 and **n** is the number of records to be sorted. If the **-q** parameter is specified, **m** is the key length plus 12.

If the **dbcdx.file.collate** runtime property is specified, it is used to translate each record for the purpose of ordering.

The sort utility returns a return code of zero if sort completes successfully. The return code is non-zero if sort does not complete successfully. If the return code is non-zero and sort is running under chain execution, then chain execution is aborted.

tdcmp

The tdcmp utility is the compiler for terminal definition files. The compiler creates **.tdb** object files from the input text files. The **.tdb** file to be used may be specified using the **dbcdx.display.termdef** runtime property.

The format of the command line is:

tdcmp *file1* [*file2*] [**-cfg=***filename*]

file1 is the name of the source text file. If the extension of *file1* is not specified, **.tdf** is assumed. This file can be any text type file. The search path for *file1* is controlled with the runtime property.

file2 is the name of the output **.tdb** file. If *file2* is not specified, *file1* with an extension of **.tdb** is the default. If *file2* is specified without an extension, **.tdb** is assumed. *file2* is created in the current directory.

-cfg=*filename* is the runtime property file.

Each record in the file consists of a function definition followed by an optional comment. The comment is preceded by a semicolon. The function definition consists of a function name followed by an equal sign (**=**) followed by a sequence of one or more characters that constitute the function. For example:

*function-name***=***function* **;** *comment*

Blanks are ignored.

function is a sequence of one or more characters that is sent to the terminal. There are several character sequences that have special meaning:

^*c* is a control character, where *c* is an upper or lower-case alphabetic character. For example, **^A** and **^a** mean control-a (decimal 1).

^[is escape (decimal 27)

#*n***#** is a one-byte value, where *n* is a one, two, or three-digit number. For example, **#27#** is the escape character.

%*v* is for the functions that use parameters. *v* is one of **h**, **v**, **t**, **b**, **l**, **r**, **a**, **n**, (**-b**), (**b-t**), (**-r**) or (**r-l**).

**** is the forcing character. It allows special characters such as **^ # % \ ;** and blank to be used literally.

There are several different types of functions. Each of the following sections describes a group of functions.

Flag Functions

no_defaults
color_erase
no_roll
scroll_erase
scroll_repos
ascii_control

The flag functions define capabilities of the terminal. These functions are not followed by an equal sign (**=**).

The **no_defaults** function means that tdcmp is not to assume the default values for **bell**, **enter_key**, **escape_key**, **backspace_key** and **tab_key** if these functions are not defined in the input file. The default values for these functions are:

If **bell** is not specified, it defaults to ^g.
If **enter_key** is not specified, it defaults to ^m.
If **escape_key** is not specified, it defaults to ^[.
If **backspace_key** is not specified, it defaults to ^h.
If **tab_key** is not specified, it defaults to ^i.

The **color_erase** function means that the terminal is capable of setting the background color during a screen erase. If this function is not defined and an erase is performed with the background color being other than **black**, then blanks are displayed to the terminal to achieve the correct background color.

The **no_roll** function means that the terminal will not roll up one line if a character is displayed in the bottom right corner of the screen. If this function not defined, then DB/C DX will try to use the **disable_roll** and **enable_roll** functions to prevent the screen from rolling up a line. Otherwise, DB/C DX will prevent any characters from being displayed in that position.

The **scroll_erase** function means that the terminal is capable of handling erase functions within scroll regions.

The **scroll_repos** function means the terminal provides cursor position recalculation within the scroll region.

The **ascii_control** function tells DB/C DX to try to use ASCII control characters to perform simple cursor movements instead of using the **position_cursor** sequence. This option may reduce character output.

Number Functions

lines
columns

The number functions define numeric capabilities of the terminal.

lines and **columns** functions specify the size of the screen. If **lines** is not specified, it defaults to 25. If **columns** is not specified, it defaults to 80.

Terminal Initialization and Terminate String Functions

initialize
terminate

The terminal initialization and terminate string functions define character strings used to initialize and reset the terminal.

initialize is a sequence of characters that is sent before the first characters are sent to the terminal. **terminate** is a sequence of characters that is sent upon termination of a DB/C DX session.

Terminal Control String Functions

position_cursor	**scroll_right**	**insert_line**
position_cursor_horz	**enable_roll**	**delete_line**
position_cursor_vert	**disable_roll**	**open_line**
clear_screen	**erase_win**	**close_line**
clear_to_screen_end	**scroll_win_up**	**cursor_underline**
clear_to_line_end	**scroll_win_down**	**cursor_block**
scroll_region_tb	**scroll_win_left**	**cursor_on**
scroll_region_lr	**scroll_win_right**	**cursor_off**
scroll_up	**insert_char**	**cursor_half**
scroll_down	**delete_char**	**special**_*nnn*
scroll_left	**bell**	

The terminal control string functions define character strings to cause the terminal to perform the corresponding function. Some of the functions support variables in the character strings. When those functions are used to perform a terminal function, the variables are replaced with formatted numeric values. The following functions support variables:

position_cursor may contain the variables **h** (horizontal) and **v** (vertical).
position_cursor_horz may contain the variable **h** (horizontal).
position_cursor_vert may contain the variable **v** (vertical).

scroll_region_tb may contain the variables **t** (top) and **b** (bottom). In addition, this function may contain the variables (**-b**) and (**b-t**) where the parenthesis are part of the variable.

scroll_region_lr may contain the variables **l** (left) and **r** (right). In addition, this function may contain the variables (**-r**) and (**r-l**) where the parenthesis are part of the variable.

erase_win may contain the variables **t** (top), **b** (bottom), **l** (left), **r** (right), **a** (attribute) and **n** (number of lines in window). In addition, this function may contain the variables (**-b**), (**b-t**), (**-r**) and (**r-l**), where the parenthesis are part of the variable.

scroll_win_up may contain the variables **t** (top), **b** (bottom), **l** (left), **r** (right), **a** (attribute) and **n** (number of lines to scroll up). In addition, this function may contain the variables (**-b**), (**b-t**), (**-r**) and (**r-l**), where the parenthesis are part of the variable.

scroll_win_down may contain the variables **t** (top), **b** (bottom), **l** (left), **r** (right), **a** (attribute) and **n** (number of lines to scroll down). In addition, this function may contain the variables (**-b**), (**b-t**), (**-r**) and (**r-l**), where the parenthesis are part of the variable.

scroll_win_left may contain the variables **t** (top), **b** (bottom), **l** (left), **r** (right), **a** (attribute) and **n** (number of columns to scroll left). In addition, this function may contain the variables (**-b**), (**b-t**), (**-r**) and (**r-l**), where the parenthesis are part of the variable.

scroll_win_right may contain the variables **t** (top), **b** (bottom), **l** (left), **r** (right), **a** (attribute) and **n** (number of columns to scroll right). In addition, this function may contain the variables (**-b**), (**b-t**), (**-r**) and (**r-l**), where the parenthesis are part of the variable.

Variables are preceded by the percent sign (**%**). The output formatting sequence comes after the variable. The first character after the variable may be a plus sign (**+**). The character following this is a one-byte value that is added to the variables before formatting. Following this is the actual output format character. Several characters have special meaning:

d means one or more ASCII characters.
2 means two ASCII characters.
c means one byte binary.
t means two ASCII characters with the following values:

0 = 0x20 0x20
1 = 0x20 0x21
...
31 = 0x20 0x3F
32 = 0x21 0x20
...
255 = 0x27 0x3F

z means two ASCII characters with the following values:

0 = 0x30 0x30
1 = 0x30 0x31
...
15 = 0x30 0x3F
16 = 0x31 0x30
...
255 = 0x3F 0x3F

n means zero or three ASCII characters. If the value is zero, then no characters are used. If the value is non-zero, then the first two characters are the same as the **z** format and the third character is **0x30**.

The **special**_*nnn* is the special screen output mapping function. When the character with the value *nnn* in decimal is sent to the output device, it is converted to the character sequence defined for that function.

Terminal Attribute String Functions

reverse_on
reverse_off
underline_on
underline_off
blink_on
blink_off
bold_on
bold_off
dim_on
dim_off
auxport_on
auxport_off
all_off
fg_black
fg_blue
fg_green
fg_cyan
fg_red
fg_magenta
fg_yellow
fg_white
fg_color_*nnn*
bg_black
bg_blue
bg_green
bg_cyan
bg_red
bg_magenta
bg_yellow
bg_white
bg_color_*nnn*
graphic_on
graphic_off
graphic_up_arrow
graphic_down_arrow
graphic_left_arrow
graphic_right_arrow
graphic_horz_line
graphic_vert_line
graphic_cross
graphic_upper_left_corner
graphic_upper_right_corner
graphic_lower_left_corner
graphic_lower_right_corner
graphic_up_tick
graphic_down_tick
graphic_left_tick
graphic_right_tick
graphic_*nnn*

The terminal attribute string functions define character strings to set terminal attributes. The attributes **reverse**, **underline**, **blink**, **bold**, **dim**, **auxport**, **graphic** have corresponding **on**/**off** functions. If an **on** function is defined but it's corresponding **off** function is not, then the function **all_off** is used to turn off the attribute. When the **all_off** function is used to turn off an attribute, it is assumed that all attributes that were currently on were turned off and the colors were reset to the terminal defaults. This will cause DB/C DX to restore the attributes and color resulting in additional terminal output. For this reason, it is better to define the **off** functions for the attributes and not rely on the **all_off** function to turn off the attribute.

fg_color_*nnn* and **bg_color**_*nnn* are the foreground and background numeric color mapping functions. The value *nnn* is a decimal number from 0 to 255. When ***color=***nnn* or ***bgcolor=***nnn* is used, the sequence **fg_color**_*nnn* or **bg_color**_*nnn* is sent to the output device. The colors ***black**, ***blue**, ***green**, ***cyan**, ***red**, ***magenta**, ***yellow**, and ***white** correspond to the numeric values 0-7 respectively. The colors combined with ***boldon** correspond to the numeric values 8-15 respectively. ***boldon** effects only foreground colors.

The graphic characters typically depend on the **graphic_on** function to cause the terminal to use the alternate character set. The **graphic_off** function reverses back to the normal character set. Some terminals may be able to support graphic characters without having to use the **graphic_on** and **graphic_off** functions. **graphic**_*nnn* is the numeric graphic character mapping function. The value *nnn* is a decimal number from 0 to 43. The graphic characters ***hln**, ***vln**, ***crs**, ***ulc**, ***urc**, ***llc**, ***lrc**, ***rtk**, ***dtk**, ***ltk**, and ***utk** correspond to the numeric values 0-10 respectively. The graphic characters combined with ***hdblon** correspond to the numeric values 11-21 respectively. The graphic characters combined with ***vdblon** correspond to the numeric values 22-32 respectively. The graphic characters combined with ***dblon** or both ***hdblon** and ***vdblon** correspond to the numeric values 33-43 respectively.

Terminal Key String Functions

enter_key
escape_key
backspace_key
tab_key
backtab_key
up_key
down_key
left_key
right_key
insert_key
delete_key
home_key
end_key

page_up_key
page_down_key
f*n*_**key** where *n* is from **1** to **20**
shift_up_key
shift_down_key
shift_left_key
shift_right_key
shift_insert_key
shift_delete_key
shift_home_key
shift_end_key
shift_page_up_key
shift_page_down_key
shift_f*n*_**key** where *n* is from **1** to **20**
control_up_key
control_down_key
control_left_key
control_right_key
control_insert_key
control_delete_key
control_home_key
control_end_key
control_page_up_key
control_page_down_key
control_f*n*_**key** where *n* is from **1** to **20**
alt_up_key
alt_down_key
alt_left_key
alt_right_key
alt_insert_key
alt_delete_key
alt_home_key
alt_end_key
alt_page_up_key
alt_page_down_key
alt_f*n*_**key** where *n* is from **1** to **20**
alt_*c*_**key** where *c* is a character from **a** to **z**
special_nnn_key

The terminal key string functions define the character sequences of the corresponding keystroke. This character sequence is the set of characters that are sent when the corresponding key is pressed. The same function may be defined more than once.

The **special**_*nnn*_**key** is the special keyboard input mapping function. Whenever the character sequence specified for this function is encountered, it is converted to the value *nnn* in decimal before it is returned as a single keystroke. The numeric value returned corresponds to the same values used by the setendkey and getendkey statements.

Here is an example of a terminal definition file:

```
lines=#24# ; number of lines on screen
columns=#80# ; number of columns on screen
position_cursor=^[y%v+\ c%h+\ c ; pos cursor
clear_screen=^[k ; *es
clear_to_screen_end=^[j ; *ef
clear_to_line_end=^[i ; *el
bell=^g
reverse_on=^[4a
reverse_off=^[4@
graphic_on=^[<a
```

graphic_off=^[<@
graphic_horz_line=@
scroll_noerase
special_130=^[<a@^[<@ ; graphics on, horz line, graphics off
special_307_key=^af^m ; function key 7
enter_key=#13#
backspace_key=#8#
escape_key=#27#
tab_key=#9#
bktab_key=^[2
insert_key=^[p\ #8#
delete_key=#127#
home_key=^[h
up_key=#11#
down_key=#10#
left_key=#30##8#
right_key=#12#
control_left_key=^[\ d
control_right_key=^[\ c
f1_key=^a@^m ; equivalent to #1#@#13#
f2_key=^aa^m
f3_key=#1#b#13# ; equivalent to ^ab^m
f4_key=#1#c#13#
f5_key=#1#d#13#
f6_key=#1#e#13#