What's PowerBSORT OLE custom control ?

PowerBSORT enables you to use high feature, high performance sort/merge programs (DLL) in OLE container applications. The PowerBSORT Dynamic Link Library (DLL) is required to perform sort/merge processes with PowerBSORT OLE custom control. PowerBSORT OLE custom control can be viewed when it's designed but cannot be displayed when it runs. You can add high performance sort/merge processes to application programs and set properties easily without much coding if you set properties after you place PowerBSORT object to forms.

Registration of PowerBSORT OLE custom control

PowerBSORT must be registered when it's designed and developed or before you run applications created with PowerBSORT OLE custom control. When both Windows 95 and Windows NT are installed on one computer, you must register it to each OS's registries. If PowerBSORT is installed by an installer, these essential registration operations are done automatically. However, if PowerBSORT OLE custom control is transferred to other directory/folder, you must register by following installation steps or registration information will not process.

Registration operation and procedures

REGSVR32.EXE (32 bits) needs to be registered by the user and may be installed to directory/folder specified by other users. The following steps show how to register OLE custom control (OCX) to the registry.

<*user_directory/folder*>\REGSVR32 \<*user_directory/folder*>\F3BEBSRT.OCX

Note:

Specify PowerBSORT OLE custom control with full path name.

When PowerBSORT OLE custom control is registered normally, a message box is displayed and informs users of the normal registration.

About the copyright

Outline of PowerBSORT

PowerBSORT is an efficient sort/merge product for business use.

PowerBSORT supports rich key formats, such as internal decimal numbers and external decimal numbers necessary for office work and also sorts a large quantity of data efficiently. Since record processes (e.g. selection/summation/reconstruction of the record) can be combined with Sort processes, you will generate processed results which meet demand.

Introduction to main features

See Also

Main features supported by PowerBSORT are sort, merge and copy features . The following help features may be classified as main features.

Sort: Sort is used to arrange records into either ascending or descending order based on key fields contained in the record.

Merge: Merge is used to combine multiple files or groups of records into a single file or group, preserving the existing order based on key fields. Records being merged must already be in order based on the same key fields that are specified to the Merge feature.

Copy: This feature copies records from one file to another. A key field is not required for Copy.

Introduction to record option features

See Also

Combining a main feature and the feature operating the record is called a record option feature. In PowerBSORT, the following five features are called record option features.

Record selection: Records are selected during Sort or Merge processing based on the contents of specified fields within the record.

Record reconstruction: During Sort , Merge or Copy processing, fields in an input record can be moved or removed, and new constant data can be added to create a new output record.

Record summation: The values of specially defined on Summary Fields are added together during Sort or Merge processing, whenever records are found to have matching keys. Only one record is output for each unique key value, and the Summary Field in that record will contain the sum of all corresponding values in the matching records.

Record Suppression: Used with Sort or Merge features, records that have keys matching a previous record are dropped. Only one record is output for each unique key value.

FIFO: When records are sorted, records with the same key values are kept in the same relative order to matching records. The first input record corresponding with each unique key value is output first, the second input record with that same value is output second.

Combination of main feature and record option feature

See Also

. The following table show which main features are combined with record option features.

Option	Sort	Merge	Copy
Record Selection	A	A	A
Record Reconstruction	A	A	A
Record Summation	A	A	NA
Record Suppression	A	A	NA
FIFO	A	NA	NA

A = Available

NA = Not Available

Supported file types

See Also

PowerBSORT supports the following files types:

- Text file
- Binary ordinary file
- Fujitsu COBOL85 sequential file
- Fujitsu COBOL85 indexed file
- Fujitsu COBOL85 relative file

Note:

Text is a document file composed of character string data. The binary fixed length file contains hexadecimal data in addition to character strings and is composed of records whose length is constant. The COBOL85 sequential file, the COBOL85 indexed file, and the COBOL85 relative file are file formats supported by Fujitsu COBOL85 and each file has a fixed length record format and variable-length record format. The following table shows the combinations of input file types and output file types. COBOL85 files indicates Fujitsu COBOL85 files.

Input File	**Text**	**Binary fixed**	**COBOL85 sequential**	**indexed**	**relative**	
Text	OK	NG	NG	NG	NG	
Binary fixed	NG	OK	OK	OK	OK	
COBOL85 sequential fixed	NG	OK	OK	OK	OK	
COBOL85 sequential variable		NG	NG	OK	OK	OK
COBOL85 indexed fixed	NG	OK	OK	OK	OK	
COBOL85 indexed variable	NG	NG	OK	OK	OK	
COBOL85 relative fixed	NG	OK	OK	OK	OK	
COBOL85 relative variable	NG	NG	OK	OK	OK	

Notes:

- If the file type is a text file, the **0x1a** code detected in the data is processed by PowerBSORT as **EOF**(End of File). Therefore, data entered after the **0x1a** of the file is not

processed. Moreover, when **0x1a** exists in an input file, the **0x1a** is added to the end of the output file. When two or more input files include **0x1a**, the size of the output file becomes smaller than the total value of the input files size because only one **0x1a** is added to the output file. If **0x1a** does not exist in the input files, **0x1a** is not added to the end of the output file.

- The record format of the output file is the same record format of the input file.

- The file type that differs from the actual file type must not be specified. Attempts to specify added data will result in program malfunction.

Data Formats

ASCII

ASCII data consists of character data with hexadecimal values from **0x00** to **0x7f**. Special processing is not required for ASCII data.

Unsigned binary

Eight bit values in a byte can be used by defining a numeric mask value that is used to select specific bits. The mask and the data byte are compared with a logical AND process (the logical product). The mask is specified as a decimal number but is translated into a binary string. For example, should the right **3** bits of a byte be desired as the key field, a mask of **7** would be specified. The **7** is translated into the binary mask of **"00000111"**. When the mask is AND'ed with the data byte, the left hand **5** bits of the product will be **0** and the right hand **3** bits will match the bits in the data byte.

Fixed point signed binary

The first bit of this binary number is assumed to be a sign. A **1** makes the number positive. A **0** is negative. All other bits are treated as a binary value.

Fixed point unsigned binary

This binary number does not have a sign bit.

IEEE format floating point binary

This **32** bit binary number uses the first bit as a sign and the next **7** bits as an exponent value from **1** to **127**. The remaining **24** bits represent the mantissa of the floating point number.

IEEE format floating point binary double precision

This **64** bit binary number uses the first bit as a sign and the next **10** bits as an exponent value. The remaining bits represent the mantissa of the floating point number.

Internal decimal number

This format, sometimes called "packed decimal", stores two decimal digits **(0-9)** in a single byte. Each digit is stored as a hexadecimal **4** bit nibble with only the valid decimal values **(0-9)** allowed. The right most nibble of the right most byte is reserved to carry the sign. A value of **0xa**, **0xc**, **0xe** or **0xf** in the sign nibble indicates positive. Values of **0xb** or **0xd** indicate negative. A **1** byte number can store **1** digit and a sign. A **2** byte number can store **3** digits and the sign. **(e.g. 0x23901c is the positive number 23901)**

External decimal

This format, sometimes called "display format", stores one decimal digit in each byte. The

left nibble **(4 bits)** of each byte is filled with **0x3**. The right nibble has the decimal digit **(0-9)**. The left nibble of the right most byte holds a sign. When this nibble is **0x4** the number is positive. **0x5** indicates negative. **(e.g. 0x34303152 is the negative decimal number -4012)**

Leading separated sign

This format is a version of display format similar to external. Decimal digits are stored in the right nibble **(4 bits)** of each byte and the left nibble is filled with **0x3**. The left most byte is reserved as a sign. A value of **0x2b** indicates positive and **0x2d** is Negative. **(e.g. 0x2b3431 is the decimal +41)**

Trailing separated sign

This format is identical to leading separated sign format except the sign byte is the right most byte. **(e.g. 0x34312b is +41)**

Leading overpunched sign

This format is identical to external decimal format except the sign is on the left nibble of the left most byte, rather than the right most byte. **(e.g. 0x54303132 is the negative decimal -4012)**

Trailing overpunched sign

This format is identical to external decimal format.

Relation between PowerBSORT data format and Visual Basic data type

The following table displays which Visual Basic data type corresponds to the data format supported by PowerBSORT

[Explanation of Key words in the following lists]

Data type: Visual Basic

Data format: PowerBSORT

Data type	Data format	1996	-1996
String	asc	3139393620	2D31393936
Byte	asc		
Integer	fbl	CC07	34F8
Long	fbl	CC070000	34F8FFFF
Single	ifl	0080F944	0080F9C4
Double	ifl	0000000000309F40	00000000000309FC0
Currency	-	C090300100000000	406FCFFFFFFFFFF
Date	-	0000000000309F40	00000000000309FC0
Boolean	-		
Variant	-		

Data type	Bytes	Range
String	1 byte par 1 character	0 to about 65,500bytes
Byte	1 byte	0 to
Integer	2 bytes	-32,768 to 32,767
Long	4 bytes	-2,147,483,648 to 2,147,483,647
Single	4 bytes	-3.402823E38 to -1.401298E-45 (negative values)
	ditto	1.401298E-45 to 3.402823E38 (positive values)
Double	8 bytes	-1.79769313486231E308 to -.94065645841247E-324 (negative values)
	ditto	4.94065645841247E-324 to .79769313486231E308 (positive values)
Currency	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Date	8 bytes	1 Jan. 100 A.D. to 31 Dec. 9999 A.D.
Boolean	2 bytes	**True** or **False**
Variant	16 or 1 byte par 1 character (in the case data is a character string)	

AlphaNumOnly Property

See Also

All key field specifications are omitted in text file sorting or merging (**DisposalNumber**=0 or 1) and **True** is to be set if alphabet, number, blank and tab are entered.

Syntax

object.**AlphaNumOnly** [= *integer*]

The **AlphaNumOnly** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
integer	Specify the value that determines behavior when key field is omitted. See "Settings" below for the value to be set.

Settings

The settings for *integer* are:

Setting	Description
True	Alphabet, number,blank and tab are entered.
False	(Default) All characters are entered.

Remarks

The value specified in this property makes sense only when text file is dealt and key field of sorting or merging is omitted. It does not make sense in other cases. This option also has the same meaning in the **KeyCmdStr** property. This operation is performed for every specified key field and **AlphaNumOnly** property is performed when all key fields are omitted.

CollationOrder Property

See Also

Key field checking order is set if all specifications of key field are omitted in sorting or merging of the text file (**DisposalNumber**=0 or 1). Checking order means how to compare character string.

Syntax

object.**CollationOrder** = {*value*}

The **CollationOrder** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
value	Specify the value that determines the key field checking order. See "Settings" below for the value to be set.

Settings

The **CollationOrder** Property settings are:

Setting	Description
0	(Default) No specification (Checking order of system).
1	Comparison is performed even if some bites exist.
2	The number including sign (+,-)character are dealt.

Remarks

The value specified in this **CollationOrder** property makes sense only when text file is dealt and key field of sorting or merging is omitted. It does not make sense in other cases. This option also has the same meaning in the **KeyCmdStr** property. This operation is performed for every specified key field and **CollationOrder** property is performed when all key fields are omitted. "Checking order of system" means Binary mode or arranging in the order of character code.

CompareAsUpperCase Property

See Also

True is set if all key field specifications are omitted in sorting or merging of the text file (**DisposalNumber**=0 or 1) and small letters are dealt as capital letters.

Syntax

object.**CompareAsUpperCase** [= *integer*]

The **CompareAsUpperCase** property syntax has the following parts:

Part	Description
object	Specify the object expression that signifies reference to the object.
integer	Specify the value that directs behavior when key field is omitted. To set up value, refer to "Settings" below.

Settings

The settings for *integer* are:

Setting	Description
True	Lowercase letters are dealt as uppercase letters.
False	(Default) Lowercase letters and uppercase letters are dealt separately.

Remarks

The value specified in this property makes sense only when text file is dealt and key field of sorting or merging is omitted. It does not make sense in other cases. This option also has the same meaning in the **KeyCmdStr** property. This operation is performed for every specified key field and **CompareAsUpperCase** property is performed when all key fields are omitted.

DispMessage Property

See Also

This property specifies if messages are to be displayed should an error be found in PowerBSORT OLE custom control execution. Set **True** if error messages are to be displayed.

Syntax

object.**DispMessage** [= *integer*]

The **DispMessage** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
integer	Specify the value determining whether error messages are visible or hidden. To set up value, refer to“ Settings” below.

Settings

The settings for *integer* are:

Setting	Description
True	Error messages are displayed when errors are found.
False	(Default) Error messages are hidden.

Remarks

DispMessage property displays error messages on the screen. The code is set to **ErrorCode**, **ErrorDetail** and **SubErrorCode** property (depending on the case) when errors are found upon execution, even if error messages are not displayed. However, you should examine the meaning of the error code from the help menu. The process is halted to display messages when errors are found if **DispMessage** property is set to display messages. If you wish to refer to error code when errors are found and continue to process (by coding to cut the process by the value), it is convenient to set the value of **DispMessage** property to **False**.

DisposalNumber Property

See Also

Main feature performed In PowerBSORT is set.

Syntax

object.**DisposalNumber** = {*value*}

The **DisposalNumber** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
Value	Specify the value that determines the main feature of PowerBSORT. To set up value, refer to “Settings” below

Settings

The **DisposalNumber** property settings are:

Setting	Description
0	(Default) Sort feature.
1	Merge feature.
2	Copy feature.

Remarks

This **DisposalNumber** property is important because sort, merge and copy features, the main features of PowerBSORT, are determined by the value specified in this property.

EnableOverwriteInputFile Property

See Also

Sets output file handling when the output file set in **OutputFile** property exists in the input file set in **InputFiles** property. Set **True** if input file can be overwritten.

Syntax

object.**EnableOverwriteInputFile** [= *integer*]

The **EnableOverwriteInputFile** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
integer	Specify the value that determines how to specify the field.

Settings

The settings for *integer* are:

Setting	Description
True	Overwriting on input file is allowed.
False	(Default) Overwriting on input file is not allowed.

Remarks

EnableOverwriteInputFile property is valid in sort process(**DisposalNumber** = 0). Therefore error occurs if **True** is set in merge process (**DisposalNumber** = 1) or copy process(**DisposalNumber** = 2).

EnableOverwriteInputFile property is a feature to protect your resource. Input file data is rewritten if the result of sorting has the same name as the input file. Therefore, do not use this value if you do not wish to input file rewrite data.

FieldDefinition Property

See Also

Sets text files field specification.

Syntax

object.**FieldDefinition** = {*integer*}

The **FieldDefinition** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
integer	Specify the value determining how to specify the field. To set up value, refer to “Settings” below

Settings

The **FieldDefinition** property settings are:

Setting	Description
0	(Default) Distinguish the field by field separation character. It is sometimes called “floating field”.
1	Specify field by column position. It is sometimes called “fixed field”.

Remarks

FieldDefinition property is valid when the processed file is text file. Files other than text file (e.g. binary file) are always calculated with the column position from the head of the record (**FieldDefinition** = 1 in this property).

How to calculate the field position of various field, such as key field(**KeyCmdStr** Property), summation field(**SumCmdStr** Property), reconstruction field (**RconCmdStr** Property) and selection field (**SelCmdStr** Property) changes depending on how field specification is set in the **FieldDefinition** property. For more information on field separation character and floating field, see **FieldDelimiter** property.

FieldDelimiter Property

See Also

Sets text file field separation character.

Syntax

object.**FieldDelimiter** = {*string*}

The **FieldDelimiter** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
string	Specify character expression used as field separation character.

Remarks

Field separation characters indicate field break points in the record. There are two ways to specify field separation characters. One option is to specify by character string and the other is to specify by hexadecimal. If you specify field separation characters by character string, enclose the whole with a quotation mark. (') If you use the quotation mark separation character, set a backslash (\) ahead of the quotation mark. If you specify with hexadecimal, insert an 'x' at the head and then set the hexadecimal code. Blank or tab code is regarded as a field separation character when they are omitted.

If you use (\) mark as a field separating character, specify two(\) marks consecutively. For example, specify (\\) if you specify(\). Do not use the same character as a record separation character.

Example of Field separation character specification

- Character string specification

  single blank: ' '

  single \mark: '\\'

  character string including quotation mark: ' \''

- Hexadecimal specification

  specification of level tub: x09

Notes on field specification divided by separation character:

- The field numbers of floating field are counted from 0.
- Separation character is not included in the field. However, blank or tab of the head of record is included in the head field if field separation character is omitted.
- The first blank becomes a separation character if blanks are continued and the rest of the blanks are regarded as a part of the field. However, an empty field is regarded as existing when separation characters continue if you specify the separation character.

FjcobAlternateKey Property

See Also

Sets index sub key when making output file Fujitsu COBOL85 index file. (**OutputFileType**= 3) The sub key defines the position and length.

Syntax

object.**FjcobAlternateKey** = {*string*}

The **FjcobAlternateKey** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
string	Specify sub key of Fujitsu COBOL85 index following the description form.

Remarks

The following is the sub key description form for **FjcobAlternateKey** property. Separate plural sub keys with /.

Description Form:

[D] (pos.len [/pos.len] ...)

D

Specify this when sub key data items overlap with other keys.

Pos(Position)

Specifies the sub key position of Fujitsu COBOL85 index file by decimal number. Relative position from the head of data item record (the bite number which starts from 0) is specified as the position.

len(Length)

Specifies the sub key length of Fujitsu COBOL85 index file by decimal number. Be sure to make the total length of main key (**FjcobPrimeKey** Property) and sub key (**FjcobAlternateKey** Property) 254 bites or less.

/

Specify plural data fields as a key by separating with '/'.

Notes:

- **FjcobAlternateKey** property operates only if Fujitsu COBOL85 file system is installed. Contact Fujitsu to obtain information about Fujitsu COBOL. (See product “About Information” for details.)

- Errors occur if output file type is not Fujitsu COBOL85 index (**OutputFileType** = 3).
- Data form of specified sub key is processed in ASCII code.

- Specified main key is arranged in ascending order. They cannot be arranged in descending order.

FjcobDataCompression Property

See Also

Specifies whether record is compressed or not when making output file Fujitsu COBOL85 index file (**OutputFileType** = 3). Set **True** to compress record.

Syntax

object.**FjcobDataCompression** [= *integer*]

The **FjcobDaraCompression** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
integer	Specify the value that directs record compression. To set up value refer to "Settings" below.

Settings

The settings for *integer* are:

Setting	Description
True	Compress record.
False	(Default) Does not compress record.

Remarks

FjcobDataCompression property is valid only when Output file type is Fujitsu COBOL85 index file (**OutputFileType** = 3). Errors occur if another type of output file is specified.

Note:

FjcobDataCompression property operates only if the Fujitsu COBOL85 file system is installed. Contact Fujitsu to obtain information about Fujitsu COBOL. (See product "About Information" for details.)

FjcobKeyCompression Property

See Also

Specifies whether to compress index key when making output file Fujitsu COBOL85 index file (**OutputFileType** = 3). Set **True** to compress index key.

Syntax

object.**FjcobKeyCompression** [= *integer*]

The **FjcobKeyCompression** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
integer	Specify the value that directs index key compression. To set up value refer to "Settings" below.

Settings

The settings for *integer* are:

Setting	Description
True	Compress index key.
False	(Default) Does not compress index key.

Remarks

FjcobKeyCompression property is valid only when Output file type is Fujitsu COBOL85 index file (**OutputFileType** = 3). Errors occur if another type of output file is specified.

Note:

FjcobDataCompression property operates only if the Fujitsu COBOL85 file system is installed. (See product "About Information" for details.)

FjcobPrimeKey Property

See Also

Sets index main key when making Output file Fujitsu COBOL85 index file (**OutputFileType** = 3). The main key is defined with the position and the length.

Syntax

object.**FjcobPrimeKey** = {*string*}

The **FjcobPrimeKey** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
string	Specify the main key of Fujitsu COBOL85 index following its description form.

Remarks

The description form of main key described to **FjcobPrimeKey** property is shown below. Specify plural main keys by separating with / .

Description Form:

[D] (pos.len [/pos.len] ...)

D

Specify this when data item sub keys overlap with other keys.

Pos(Position)

Specifies the sub key position of Fujitsu COBOL85 index file with a decimal number. Relative position from the head of data item record (the bite number which starts from 0) is specified as the position.

len(Length)

Specifies the main key length of Fujitsu COBOL85 index file with a decimal number. Make the total of length of main key (**FjcobPrimeKey** Property) and sub key (**FjcobAlternateKey** Property) 254 bites or less.

/

When you specify plural data fields as a key, separate with '/'.

Notes:

- **FjcobPrimeKey** property operates only if the Fujitsu COBOL85 file system is installed. (See product "About Information" for details.)

- Errors occur if output file type is not Fujitsu COBOL85 index (**OutputFileType** = 3).

- Data form of specified sub key is processed in ASCII code.

- Specified main key is arranged in ascending order. They cannot be arranged in

descending order.

HandlingSameKey Property

See Also

Sets how to process some key fields having the same contents. It does not process when the copy feature is used (**DisposalNumber** = 2).

Syntax

object.**HandlingSameKey** = {*value*}

The **HandlingSameKey** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
value	Specify the value determining how to process when key field is the same. To set up value refer to "Settings" below.

Settings

The **HandlingSameKey** property settings are:

Setting	Description
0	(Default) Processes indefinitely.
1	Determines the order of record output with FIFO.
2	Deletes record suppressing.
3	Adds summary field with record summary feature.

Remarks

Compares records following key field setting in **KeyCmdStr** property. If identical records are found **HandlingSameKey** property directs how to process them. Therefore, the FIFO feature, suppress feature and record summary feature will not operate alone.

Features provided in **HandlingSameKey** property have exclusive relation to each other.

The following is the explanation of processes (terms) used in **HandlingSameKey** property.

FIFO feature:

When sorting a file containing some records with the same value key field, this feature outputs records to retain their original order. This feature operates in the sort feature (**DisposalNumber** = 0).

Suppress feature:

When you sort or merge a file containing some records with the same value key field, this feature deletes all other records leaving only one record. The suppress feature operates in the sort/merge feature (**DisposalNumber** = 0 or 1).

Record summary feature:

When you sort or merge a file containing some records with the same value key field, this feature adds the values of the summation field (the field set in **SumCmdStr** property) and makes one record. This feature operates in the sort/merge feature (**DisposalNumber** = 0 or 1).

IgnoreControlCode Property

See Also

All key fields specifications are omitted in sorting or merging of the text file (**DisposalNumber** = 0 or 1). Set **True** to ignore control code.

Syntax

object.**IgnoreControlCode** [= *integer*]

The **IgnoreControlCode** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
integer	Specify the value directing whether control code is ignored or not. To set up value refer to "Settings" below.

Settings

The settings for *integer* are:

Setting	Description
True	Ignore control code.
False	(Default) Process control code.

Remarks

The value specified in this Property operates only under the following conditions:

- The processed file is a text file.
- Key field of sorting or merging is omitted.

This property will not operate on other conditions. Use caution. There are other options that ignore control code in **KeyCmdStr** property. This operation is used for every specified key field and **IgnoreControlCode** property operates only if all key fields are omitted.

InputFiles Property

See Also

Sets Input files path name to perform sorting, merging and copying.

Syntax

object.**InputFiles** = {*string*}

The **InputFiles** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
string	Specify Input files path name.

Remarks

Input file path names described in **InputFiles** property support long file names. Plural Input file path names may also be set. File name rules are described in the system. Use the following specifications to set plural Input files.

Note:

Placing blank(s) between file names is regarded as a separation. Therefore, if you want to set a file name including a blank, enclose it with a double quotation.

InputFileType Property

See Also

Sets Input file type to perform sorting, merging and copying.

Syntax

object.**InputFileType** = {*value*}

The **InputFileType** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
value	Specify the value determing Input file type. To set up value refer to "Settings" below.

Settings

The **InputFileType** property settings are:

Setting	Description
0	(Default) Text file.
1	Binary fixed length file.
2	Fujitsu COBOL85 sequential fixed length file.
3	Fujitsu COBOL85 sequential variable length file.
4	Fujitsu COBOL85 index fixed length file.
5	Fujitsu COBOL85 index variable-length file.
6	Fujitsu COBOL85 relative fixed length file.
7	Fujitsu COBOL85 relative variable-length file.

Remarks

InputFileType property sets the file type to be processed. This property can set only one type of file. Therefore, plural InputFiles set in **InputFiles** property must be the same file type.

Note:

This property operates only if Fujitsu COBOL85 file system is installed, even if Fujitsu COBOL85 file is set (the option numbers 2-7 are set) as InputFile Type. (See product "About Information" for details.)

KeyCmdStr Property

See Also

Sets sorting and merging key fields. The key fields define the position, length, data form and key operation. The entire record is entered as ASCII code key field if it is omitted.

Syntax

object.**KeyCmdStr** = {*string*}

The **KeyCmdStr** property syntax has the following parts:

Part	Description
object	Specify the object expression that refers to the object.
string	Specify key field following its description form.

Remarks

The following is the key field description form described to **KeyCmdStr** property. When specifying plural key fields, enter them continuously or separate with a comma.

Description Form :

pos.len typ opt [pos.len typ opt ...]

pos(Position)

The key field position is specified with a decimal number. Calculate the position regarding the head of record as 0. If text file is set in **InputFileType** property, whether it is the position of field or column is decided depending on the value that was set in **FieldDefinition** property and the meaning may change. Calculation is performed as the position of column if **InputFileType** property is set to any file other than text file.

len(Length)

The key field length is specified with a period (.). followed by a decimal number. When an unsigned binary number is specified, enter the mask value for length with a decimal **1-255**. The logical product of the field value and the mask value then become the key values. For example, when the field value is **0x8e** and the mask value is 3(**0x03**), the key value is **0x02**. When a field longer than the specified length appears for floating field text, the fields are processed using the specified field length.

typ(Type or Data Format)

Key field data format is specified immediately after **len**. For Data Format specification, see "Key field data Format and the length".

opt(Option)

Key field order is specified immediately after **typ**. For more information, see "Order of key field".

Key Field Data Format and length

Data Format	typ	Length
ASCII code	asc	1 to record length
EBCDIC code	ebc	1 to record length
Unsigned binary (bit)	bit	1 to 8bits
Signed Fixed point binary	fbi	1 to 256
Unsigned fixed point binary	ufb	1 to 256
8086 format fixed point binary	fbl	1 to 256
Unsigned 8086 format fixed point binary	ufl	1 to 256
IEEE format floating point binary	ifl	1 to 256
Internal decimal	pdl	1 to 256
Unsigned internal decimal	pdu	1 to 256
External decimal	zdl	1 to 256
Unsigned external decimal	zdu	1 to 256
Leading separate signed number	als	2 to 256
Trailing separate signed number	ats	2 to 256
Leading overpunch signed number	alo	1 to 256
Trailing overpunch signed number	ato	1 to 256

Note:

ASCII and EBCDIC code may be used with text and binary files. Other data formats may only be used with binary files.

Order of key field

Except the text file

[a | r]

Order	Meaning
a	Arrange in ascending order.
r	Arrange in descending order.

Note:

"a" and "r" cannot be specified simultaneously. When "a" and "r" are omitted, "a" is specified.

At the text file

[a | r] [b] [d] [i] [j] [n | w]

Order	Meaning
a	Arrange in ascending order.

b	Ignore leading blanks and tabs.
d	Only blanks and tabs, alphanumeric characters are compared. (Note1)
i	Ignore control character codes (unprinted character).
j	Compare lower-case letters as upper-case letters.
n	Numeric values which contain a sign are compared as arithmetic values.
r	Arrange in descending order.
w	Compare and arranges key field even if they include characters consisting of plural bites.

Notes:

- "a" and "r" cannot be specified simultaneously. When "a" and "r" are omitted, "a" is specified.
- "w" can be specified only when the type of Data Format is ASCII code.
- "n" and "w" are exclusive features. They cannot be specified simultaneously.
- When specifying plural orders, describe them continuously.

Note1:

Blanks and tabs are not recognized as control characters.

For example:

(1) 0.10asca

The 10 byte ASCII field at the beginning of the record is sorted in ascending order.

(2) 0.5ascr,20.1zdla

The 5 byte ASCII field at the beginning of the record is sorted in descending order and the 1 byte external decimal number in byte 20 is sorted in ascending order.

LineDelimiter Property

See Also

Set up separate characters in a text file record.

Syntax

object.**LineDelimiter** = {*value*}

The **LineDelimiter** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.
value	Specify the numerical value distinguishing separate record characters. For value set up, see “Settings” below.

Settings

The **LineDelimiter** property settings are:

Setting	Description
0	(Default) CRLF
1	CR
2	LF

Remarks

Separate record characters can be selected among CRLF (Carriage Return and Line Feed), CR (Carriage Return) and LF (Line Feed). A common format text file is separated by CRLF. CR and LF use 1 byte to separate record characters. CRLF uses 2 bytes. Use special attention when calculating text file record length.

MaxRecordLength Property

See Also

Set up a record length or maximum record length to be the decimal integer. Using a text file, set up the maximum record length including the line feed code (the separate characters on a record) explained in **LineDelimiter** property.

Syntax

object.**MaxRecordLength** = {*value*}

The **MaxRecordLength** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.
value	Specify the numerical value defining separate record characters. For value set up, see "Settings" below.

Settings

The **MaxRecordLength** property settings are:

Setting	Description
1 to 32,000	Record length

Remarks

Make the upper bound **30,720** bytes as a standard. The upper bound of the maximum record length can be fixed up to **32,000** bytes. Should an error message appear after fixing the maximum length at **30,720** bytes, check the total length of the specified Key field, the field length reorganized by the record reorganization feature, and the field length selected by the record selecting feature to verify you have not exceeded the processing limit of this program. Correct the problem and execute program again. Should you continue to experience problems, try reducing the number of key fields and process twice by dividing the key fields. At the text file, correct the maximum record length by including separate characters on each record.

Notes: If you fix a record length, note the following points:

- The maximum record length is a value with numbers of bytes up to a line feed and the numbers of bytes of separate characters on a record are added.
- Character of two-byte code (such as an em-size Japanese character) is calculated by one character numbering two bytes.
- If a longer record than the fixed maximum record length exists, stop processing.

OutputFile Property

See Also

Set up a file path name which outputs sort, merge, and copy processing results.

Syntax

object.**OutputFile** = {*string*}

The **OutputFile** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.
string	Specify an output path name.

Remarks

An output path name described in **OutputFile** property supports a long file name. It cannot set up two or more file path names, unlike the **InputFiles** property. File name rules are described in the Windows system. When setting up a file name with a space, add double quotation marks.

OutputFileType Property

See Also

Set up a file type which outputs of sort, merge, and copy processing results.

Syntax

object.**OutputFileType** = {*value*}

The **OutputFileType** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.
value	Specify a numerical value to the output file type. For value set up, see "Settings" below.

Settings

The **OutputFileType** property settings are:

Setting	Description
0	(Default) Text file
1	Fixed length of binary file
2	Sequential organization file of Fujitsu COBOL85
3	Index file of Fujitsu COBOL85
4	Relativity file of Fujitsu COBOL85

Remarks

OutputFileType property sets up an output file type. Note only one file type can be set up by this property.

Note:

Fujitsu COBOL85 file system operates only when installed, even if it is set up (Setting option numbers 2-4 up to value) as an output file type. Contact Fujitsu to obtain information about Fujitsu COBOL. (See product "About Information" for details.)

RconCmdStr Property

See Also

Set up reconstruction field. The reconstruction field defines the position, length or user predetermined value, its length and data format.

Syntax

object.**RconCmdStr** = {*string*}

The **RconCmdStr** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.
string	Specify reconstruction field according to its description format.

Remarks

Record reconstruction feature changes the position of a field to embed user-defined values into record. It combines the processes of sorting, merging and copying. When reorganizing a record, specify the field you wish to use sequentially from the left end of the output record. If you wish to use the field of input record, specify its position and length. If you want to use a self-defined value, specify the value.

In addition, if you use the record reconstruction feature, pay special attention to setting up the **KeyCmdStr** property and the **SumCmdStr** property. When the record reconstruction feature is used, these property values specify position and length against the form of output record to be reconstructed.

The following is the description format of a record reconstruction feature described in the**RconCmdStr** property. When you specify two or more reconstruction fields, separate them by inserting a comma (,) .

Description Form:

{ pos.len | slf.len typ } [,{ pos.len | slf.len typ } ...]

pos.len

Specify the reconstruction field position and length. A period is put between "pos" and "len" as shown in the above example. If a specified field is outside the record, an error occurs.

pos(Position)

Specify the reconstruction field position with a decimal number. Calculate the head of the record as **0**. If a text file is set up with the an input file type (**InputFileType** Property), whether it is on a field position or in a column position depends on the value set up by the **FieldDefinition** property. This property will determine its position and affect its meaning. Moreover, when the input file type (**InputFileType** property) is set up with a text file, it is calculated using the column position.

Note:

If the specified field does not exist with the input record, an error occurs.

len(Length)

Specify the reconstruction field length with a period (.) and a decimal numeral. When specifying a binary numeral (**bit**) without a mark to a data format, make the mask value from **1** to **255** decimal numbers in length.

In this example, the key value is conjugation of the field value and the mask value. For instance, when the field value is specified as **0x8e** and the mask value as **3(0x03)**, the key value is **0x02**. A floating field is a longer field than the specified field length and is processed with the specified field length. When a shorter field than the specified field length is displayed, it is processed with its actual field length.

slf.len typ

The specification method to pad a self-defined value to an output record is accomplished by placing a period between "slf" and "len". See prior example.

slf(Self-defined Values)

Specify the self-defined value to pad an output record. Self-defined values are specified by the following three methods:

- Character string values: '*character string*' (e.g. '123', 'AbCD')

  Enclose the character string with quotation marks.

- Hexadecimal numbers: x###### (e.g. x313233)

  Put an "X" at the head and continue the hexadecimal number code.

- Decimal numbers: d#### (e.g. d123, d+123, d-123)

  Put a "d" at the head and continue the decimal numbers. Symbols "+" and "-" may also be added.

Only characters can be used in a text file.

len(Length)

Specify the self-defined value length with a period (.) and a decimal number.

typ(Type or Data Format)

Specify the self-defined value data format.

Self-defined Value of a reconstruction field

A self-defined value is a constant or a character string constant of a decimal number or a hexadecimal number. For instance, the self-defined value is used to add the decimal number 00 value to a output record field or is used for other purposes. A self-defined value is determined by a value, a format and a length.

Notes of self-defined value

A character equal to a record separation character cannot be specified as a self-defined value. In addition, a floating field equivalent to a record separation character may not be specified.

- Marks can be defined on specification of a decimal number.

- When you specify an quotation mark (') , specify it sequentially ('').
- If the value (slf) specified by an self-defined value is not as long as the length specified by the length (len), use the corresponding processes below.

When the length of a self-defined value is shorter than the "len".

Character string: Add a self-defined value to the left, then add spaces to the right blank.

Figure: Add a self-defined value to the left, then add 00 to the left blank.

When the length of a self-defined value is longer than the "len".

Character string: Add a self-defined value to the left and ignore the remainder.

Figure: Add a self-defined value to the right and ignore the remainder.

Note:

If a self-defined value described with a hexadecimal number is an odd number, an error occurs because the value of hexadecimal number should consists of a 2 bytes unit. For instance, when a self-defined value is specified as x234.1asc, it becomes an ASCII code whose length (len) is 1 byte even though the specified self-defined value is a 3 digit odd number. In this case, the specified self-defined value cannot recognize the ASCII code because 1 byte in the ASCII code consists of a 2 digit number.

Data format of a reconstruction field and length of a self-defined value

The following shows the specified reconstruction field data format and the self-defined value lengths.

Except the text file

Type of Data format	typ	Length	Data format that can be specified
ASCII code	asc	1 to 256	Character or Hexadecimal
Signed fixed point binary	fbi	1 to 8(Note)	Decimal
Unsigned fixed point binary	ufb	1 to 8(Note)	Decimal
Internal decimal	pdl	1 to 16	Decimal
External decimal	zdl	1 to 18	Decimal

At the text file

Type of Data format	typ	Length	Data format that can be specified
ASCII code	asc	1 to 256	Character or Hexadecimal

Note:

Fixed point binary numbers and fixed zero point unsigned binary numbers can be specified within 8 bytes. The fixed zero point binary numbers range can be specified from -2147483647 (0x80000001) to 2147483647(0x7fffffff). On the other hand, the unsigned binary number range can specify from 0(0x00000000) to 4294967295(0xffffffff).

For example:

(1) 20.10,50.12,30.22

Reorganize the input record sequentially following the fields shown below and then output the record.

First field : 10 bytes from the 20th byte of the input record.

Second field: 12 bytes from the 50th byte of the input record.

Third field : 22 bytes from the 30th byte of the input record.

(2) 'abc'.8asc,20.10

Reorganize the input record sequentially following the fields shown below and then output the record.

First field : Enbed 8 bytes into the character string 'abc' of an ASCII code.

Second field: 10 bytes from the 20th byte of the input record.

Reverse Property

See Also

For text file sorting or merging processes(**DisposalNumber** = 0 or 1) set up the row when all key field specifications are omitted, **False** is an ascending order and **True** is a descending order. When these specifications are omitted, ascending order is set up.

Syntax

object.**Reverse** [= *integer*]

The **Reverse** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.
integer	Specify a value indicating the row default key fields. For value set up refer to "Settings" below.

Settings

The settings for *integer* are:

Setting	Description
True	Sort in descending order.
False	(Default) Sort in ascending order.

Remarks

The value specified in this property has meaning only when the key field processed both by sorting and merging is omitted, and the file to be processed is a text file. If these cases are not met, the value is meaningless. Similarly, sorting operation exists in the **KeyCmdStr** property option. It operates specified key fields sequentially. **Reverse** property, however, is a operation available only when all key fields are omitted.

SelCmdStr Property

See Also

Set up the selection field. The record selection field defines the selection condition by the logical expression.

Syntax

object.**SelCmdStr** = {*string*}

The **SelCmdStr** property syntax has these parts:

Part	Description
object	Specify an object expression that refers to the object.
string	Selection field is specified according to the description format.

Remarks

This record selection feature specifies the method of selecting a record to be processed and uses only necessary records. The record selection feature combines sorting, merging or copying processes. If you select a record, specify a compared field, a comparing field and comparison operator, or specify a compared field, a self-defined value and a comparison operator. This feature then compares two specified fields according to the comparison operator and determines whether to or not to select the sorted, merged, and copied record.

The selection field description format described in **SelCmdStr** property is as follows; When specifying two or more selection fields, tie each selection together using **AND** or **OR.**

Description Format:

pos.len typ opt.cmp. { pos.len typ | slf } [{ AND | OR } pos.len typ opt.cmp. { pos.len typ | slf } ...]

Place a period (.) between **pos** and **len** and describe continuously between **len** and **typ**. Put periods in front and behind **cmp**, and describe continuously without using blank(s).

pos(Position)

Specify the position of a selection field by a decimal number. Calculate the head of the record as **0**. If a text file is set up with the input file type (**InputFileType** Property), whether the specified position is a field position or a column position, it is judged by the value set up in **FieldDefinition** property and its meaning changes. Moreover, when the input file type (**InputFileType** property) is set with a text file, **pos** is calculated at the column position.

Note:

An error occurs when the specified field is outside the record.

len(Length)

Specify the selection field length by a period and a decimal number. When specifying an unsigned binary number (**bit**) for a data format **typ**, use the mask value to be a decimal number from 1 to 255. In this case, conjugation of the field value and the mask value reaches the key value. For instance, when the field value is specified as the hexadecimal number **0x8e** and the mask value as **3(0x03)**, the key value becomes hexadecimal number **0x02**. When a floating field is a longer field than the specified field length, it is processed by the specified field length. A shorter floating field than the specified field length is processed with its actual field length.

typ(Type or Data Format)

Specify the selection field data format. For details on the data format specifications, refer to "data format of the compared field and the comparing field" . For data format length specifications, refer to "data format and its length".

opt(Option)

Specify the selection field operation. This specification is effective when the input file type is a text file. The operation instruction is shown below.

Options	Meaning
b	Disregard the first blank of the key field.
d	Compare only blank and alphanumeric characters.
i	Disregard a code of control characters.
j	Compare English small letters as an English capital letter.
n	Compare the character string of figures which includes a mark with an arithmetic value.

cmp(The comparison operator)

Specify a comparison operator. Comparison operators are listed below.

Comparison Operator	Meaning (true case)
eq	compared field = comparing field (including a self-regulated value)
ne	compared field != comparing field (same)
gt	compared field > comparing field (same)
ge	compared field >= comparing field (same)
lt	compared field < comparing field (same)
le	compared field <= comparing field (same)

slf(Self-defined Values)

If you compare values using a self-defined value, specify the formats listed below. For data specified as a self-defined value, refer to "data format specified for a self-defined value".

- Character string values: 'character string' (e.g. '123', 'AbCD')
- Hexadecimal numbers: x###### (e.g. x313233)
- Decimal numbers: d#### (e.g. d123, d+123, d-123)

Only characters can be used as a text file.

Data format of a compared field and a comparing field

The data formats specified for a compared field and a comparing field are shown below.

Except for a text file

COMPARED FIELD	COMPARING FIELD
ASCII code	ASCII code
Unsigned binary (bit)	Unsigned binary (bit)
Fixed point binary	Fixed point binary
Unsigned fixed point binary	Unsigned fixed point binary
Internal decimal	Internal decimal
	External decimal
External decimal	Internal decimal
	External decimal
Leading separate signed number	Leading separate signed number
	Trailing separate signed number
Trailing separate signed number	Leading separate signed number
	Trailing separated sign
Leading overpunch signed number	Leading overpunch signed number
	Trailing overpunch signed number
Trailing overpunch signed number	Leading overpunch signed number
	Trailing overpunch signed number

At a text file

COMPARED FIELD	COMPARING FIELD
ASCII code	ASCII code

Data Format specified for a Self-defined Value.

Compared fields and the data formats specified for a self-defined value are shown below.

Except for a text file

COMPARED FIELD	SELF-DEFINED VALUE
ASCII code	Character or Hexadecimal

Unsigned binary(bit)	Decimal
Fixed point binary	Decimal
Unsigned fixed point binary	Decimal
Internal decimal	Decimal
External decimal	Decimal
Leading separate signed number	Decimal
Trailing separate signed number	Decimal
Leading overpunch signed number	Decimal
Trailing overpunch signed number	Decimal

At the text file

COMPARED FIELD	SELF-DEFINED VALUE
ASCII code	Character or Hexadecimal

The data format and its length

The followings are data formats and its length.

Except the text file

Data Format	typ	Length(bytes)
ASCII code	asc	1 to 256
EBCDIC code	ebc	1 to 256
Unsigned binary (bit)	bit	1 to 8 bits
Fixed point binary number	fbi	1 to 256
Unsigned ,†ixed point binary	ufb	1 to 256
Internal decimal number	pdl	1 to 256
External decimal number	zdl	1 to 256
Leading separate signed number	als	2 to 256
Trailing separate signed number	ats	2 to 256
Leading overpunch signed number	alo	1 to 256
Trailing overpunch signed number	ato	1 to 256

At a text file

Data Format	typ	Length (bytes)
ASCII code	asc	1 to 256

For example:

(1) 20.10asc.eq.30.10asc

Compare the ASCII code of 10 bytes from the 20th byte of the input record and the ASCII code of 10 bytes from the 30th byte of the input record. If both codes are equal in length, make the record to be processed.

(2) 20.10asc.ne.'abcd'

Compare the ASCII code of 10 byte from the 20th byte of an input record and the character string 'abcd' in the ASCII code. If both codes are equal in length, make the record to be processed.

(3) 12.4fbi.ge.d30

The record whose length is 30 bytes or more in decimal number is made to the record to be processed by the field of 4 byte length in fixed point binary number from the 12th byte of the input record.

Descriptions Using AND or OR

20.10pdl.lt.d123 AND 50.4zdl.gt.d-123

For input records, select records corresponding to both following conditions.

Condition 1: The field of 10 byte length in internal decimal number from the 20th byte of an input record is shorter than decimal number 123.

Condition 2: The field of 4 byte length in external decimal number from the 50th byte of an input record is longer than decimal number -123.

"20.10asc.eq.'abcd' OR 50.4zdl.gt.d123"

For input records, select records corresponding to either of the following two conditions:

Condition 1: The record including the character string "abcd".

The position of the character string inside the record is at the 20th byte from the head of the record, and its length is 10 bytes.

When specifying "abcd" without a blank, as shown above, the blank of 6 characters is set up after 'd' so it is a 10 byte length.

Condition 2: The record including decimal number above 123.

The position of the number inside the record is at the 50th byte from the head of the record, and its length is 4 bytes.

Note:

The priority level **AND and OR** becomes in the order of **AND > OR.** Parentheses or brackets cannot be used. Write the logical expression as **(a AND b) OR(c AND d)** to read **a AND b OR c AND d**, the logical expression **a OR (b AND c)** to read **a OR b AND c** and the logical expression **a AND (b OR c)** to read **a AND b OR a AND c.**

SkipLeadingBlank Property

See Also

In text file sorting or merging processes (**DisposalNumber** = 0 or 1) all key field specification is omitted and **True** is set when the first blank and the tab are disregarded.

Syntax

object.**SkipLeadingBlank** [= *integer*]

The **SkipLeadingBlank** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.
integer	Specify the value indicating whether the first blank and the tab are disregarded or not. For value set up, refer to "Settings" below.

Settings

The settings for *integer* are:

Setting	Description
True	Disregard the first blank and the tab.
False	(Default) Process all characters

Remarks

The value specified in this property has meaning only when the file to be processed is a text file and the key field of sorting and merging processes are omitted. Note it does not have meaning in other cases. Similarly, what operates this order also exists in the option <**KeyCmdStr** property>. Note the difference is the operation of each specified key field. **SkipLeadingBlank** property is an operation when all key fields are omitted.

SumCmdStr Property

See Also

Set up the Summation field when sorting or merging processes are completed. The summation field defines the position, length, and the data format. The summation field value doubles the key field value being set up in **KeyCmdStr** property but cannot be specified.

Syntax

object.**SumCmdStr** = {*string*}

The **SumCmdStr** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.
string	Specify a summation field according to its description format.

Remarks

The summation feature adds the summation field being set up in the **SumCmdStr** property one after another when all the key field values specified in **KeyCmdStr** property possess the same records. However, even if the summation field is set with **SumCmdStr** property, records are not summed unless the value of **HandlingSameKey** property is 3 (record summation).

Note:

The record summed and output cannot be made to a specific record. The remaining records are undecided and then decided arbitrarily by the execution condition and the environment of PowerBSORT. Moreover, when an overflow occurs during summation processing, it is interrupted at that time, but the processing of PowerBSORT is continued. If the error is not detected by the processing afterwards, the **0** shows a normal end is set in **ErrorCode** property. To determine whether an overflow occurred, use value **ErrorDetail** property to reflect code **115**.

The key field description format described in **SumCmdStr** property is the following; When specifying two or more summation fields, specify a key field continuously or specify them with a comma (,).

Description Format:

pos.len typ [pos.len typ ...]

pos(Position)

Specify the summation field position by decimal number. Calculate the head of the record as **0** at the position.. If a text file is set up by the input file type (**InputFileType** Property),

whether the specified position is a field position or a column position, it is judged by the value set up with the **FieldDefinition** property, changing its meaning. Moreover, when the input file type (**InputFileType** Property) is set with a text file, the position of a summation field is calculated at the column position.

len(Length)

Specify the summation field length by a period (.) and a decimal number. If you specify a unsigned binary number (**bit**) for data format **typ**, specify the mask value to be a decimal number from **1** to **255**. In this case, conjugation of the field value and the mask value reaches the key value. For example, if the field value is specified as hexadecimal **8e** and the mask value is specified as decimal **3**, the key value becomes hexadecimal **02**.

Note:

If you set up a reconstruction field using the **RconCmdStr** property, specify the summation field according to the field shown by the reconstruction field.

When the floating field is processed with a text file, the summation result is processed by the specified field length. Conversely, when a field shorter than the specified field appears, the summation result is calculated by the actual field length. See the following examples.

How are floating fields summed ?

It is assumed that following records are summed.

Record 1: F5

Record 2: F123

When the summation field is 0.1asc, 5 and 1 are added and the result of summation becomes 6.

When the summation field is 0.3asc, 5 and 123 are added and the summation result becomes 128.

typ(Type or Data Format)

Specify the summation field data format to follow **len**. For specifying data format, refer to " summation field data format and its length".

Summation field data format and its length

The summation field data format and its length are shown as follows.

Except for a Text File

The data format	typ	Length (number of bytes)
Internal decimal number	pdl	1 to 16
External decimal number	zdl	1 to 18

Fixed point binary	fbi	1 to 8
Unsigned fixed point binary	ufb	1 to 8

At a Text file

The data form	typ	Length (number of bytes)
ASCII	asc	1 to 256

For example

(1) 20.10zdl

Sum the field of 10 bytes in external decimal number from the 20th byte of the input record.

(2) 10.10zdl30.8fbi

Sum the field of 10 bytes in external decimal number from 10th byte of the input record and the field of 8 bytes in fixed point binary number from 30th byte of the record.

Notes:

- Complete Summation fields must be included within a record
- Specify a summation field so that it does not overlap a key field or another summation field
- The remaining records summation results are undecided
- If an overflow occurs during summation fields addition process, summation process is not completed
- Numbers containing a decimal point cannot be summed

TempDir Property

See Also

Set up an temporary file allocated directory name used in sorting processes.

Syntax

object.**TempDir** = {*string*}

The **TempDir** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.
string	Specify the temporarily file directory name according to the description format.

Remarks

During sorting process, if the data amount is excessive, data may not be processed due to memory restraints. A temporary file collects unprocessed data.

TempDir property sets the directories (or the folder names) to create the temporary file. In **TempDir** property, two or more directories (or folder names) can be specified in **TempDir** property.. If two or more directories (or folder names) are specified, errors result due to insufficient free space. When specifying two or more directories (or folder names), specify a different drive using a semicolon (;).When the directory (or file) name of a temporary file is omitted, create temporary files according to the following priority:

1. The directory or folder specified by the environmental variable **BSORT_TMPDIR**.
2. The directory or folder specified by the environmental variable **TEMP**.
3. The directory or folder specified by the environmental variable **TMP**.
4. Windows system directory or folder.

Note:

If the specified directory (or the folder name) does not exist, PowerBSORT displays an error message.

UsableMemorySize Property

See Also

Specify the memory size PowerBSORT uses in a decimal number integer. The setting is treated as a unit of **kilobyte** (1024 bytes).

Syntax

object.**UsableMemorySize** = {*value*}

The **UsableMemorySize** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.
value	Specify the numeric of the memory size. For value set up, refer to "Settings" below.

Settings

The **UsableMemorySize** property settings are:

Setting	Description
64 to 32,767	(Default = **0**) memory size (KB, kilobyte).

Remarks

When the value of **UsableMemorySize** property is a default, the value **0** is set up with **UsableMemorySize** property. This value **0** has a special meaning and is operated by the memory size PowerBSORT decides. PowerBSORT can operate with the value of **64** kilobytes or more. Therefore, if the value is set up as a number **1-63** with **UsableMemorySize** property, an error occurs.

PowerBSORT uses specified memory size. However, the specified memory size may not be secured by a particular type of environment at execution time. The processing is continued with the memory size secured in the execution time. However, when an obstacle occurs within execution, set up the error detail code showing memory insufficiency to **ErrorCode** property and interrupt the processing. At this time, enlarge the specified memory size and execute the processing again. Should memory errors occur (even if the memory size is enlarged), check the execution environment where other application programs use memory. If other applications are open, close them to free up memory.

Generally, the more the memory size set up with **UsableMemorySize** property, the faster its processing is executed. However, greater performance might not be demonstrated due to constant memory swapping, caused when the memory size is too large. To best utilize the operation environment adjust memory accordingly.

ErrorCode Property

See Also

The return code of PowerBSORT is notified with the LONG value. Reference to this property is available only when PowerBSORT OLE (Custom Control) is executed.

Syntax

object.**ErrorCode**

The **ErrorCode** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.

Values to be returned

The values to be returned to **ErrorCode** property are:

Error code	Description
0	Terminated Successfully.
-1	A error was detected while processing PowerBSORT DLL.
-2	A error was detected while the syntax analysis processing with PowerBSORT OLE (Custom Control).
-3	A error was detected while the syntax analysis processing with PowerBSORT OLE (Custom Control). (This is a peculiar error detected only in PowerBSORT OLE).
-4	The error in PowerBSORT OLE (Custom Control) was detected.

Remarks

ErrorCode property notifies error code when an error is detected during PowerBSORT execution time processing. Refer to **ErrorCode** property for initial problem solving.

ErrorDetail Property

See Also

Error detail codes of PowerBSORT are notified with the LONG value. Reference to this property is possible only when PowerBSORT OLE (Custom Control) is executed.

Syntax

object.**ErrorDetail**

The **ErrorDetail** property syntax has the following parts:

Part	Description
object	Specify a object expression that refers to the object.

Remarks

ErrorDetail property notifies detailed code when an error is detected during PowerBSORT execution time processing. Refer to **ErrorDetail** property for initial problem solving. For notified values, refer to “error details code list”.

SubErrorCode Property

See Also

The error code detected by the file system utility supported by PowerBSORT is notified with the LONG value. Reference to this property is possible only when PowerBSORT OLE (Custom Control) is executed.

Syntax

object.**SubErrorCode**

The **SubErrorCode** property syntax has the following parts:

Part	Description
object	Specify an object expression that refers to the object.

Remarks

SubErrorCode property notifies detailed code when an error is detected during PowerBSORT execution time processing. Refer to **SubErrorCode** property for initial problem solving. For notified value, refer to “error code concerning Fujitsu COBOL85 index file“.

Action Method

See Also

Call the DLL of PowerBSORT.

Syntax

object.**Action**

Remarks

There is no argument in the **Action** method. You must call DLL of PowerBSORT based on the value specified with various properties which PowerBSORT OLE (Custom Control) offers. Be sure to set up the value to the specified property before using the **Action** method.

Error detail codes(ErrorCode = -1 or -2)

See Also

Values of ErrorDetail property when ErrorCode property is -1 or -2:

Code	Explanation
50	The specified memory size is too small.
52	Memory necessary for PowerBSORT could not be allocated.
56	Error in the specification relates to files information. 1) The input file record format does not match the specified record format. 2) The binary file size is not a multiple of the record length. 3) Overwrite for Merge or Copy.
57	Mutually exclusive parameters were specified. 1) First In First Out (FIFO), suppress, and summation. 2) Sort, Merge, and Copy.
59	Error in the Key field 1) The key field points are outside the record. 2) The format of the key is invalid. 3) The key length is outside of support.
60	Error in the summation field 1) The summation field points are outside the record. 2) The format of the summation key is invalid. 3) The length of the summation key is outside of support.
61	The key field and the summation field or two or more of the summation fields overlap.
62	An unsupported feature was specified.
63	Error in the selection field.
64	Error in the reconstruction field.
65	Error in the file system specification.
66	The selection field does not exist in the input record.
67	The reconstruction field does not exist in the input record.
111	The mistake is found in the length of the record.
114	The input files merge processing is not sorted.
115	An overflow occurred in summation processing.
116	A variable-length or text record input did not include the summation field. Summation processing was halted but other processing was continued.
117	There is no line Delimiter character on the text record.
118	A text record input did not include the key field.
200	A read error occurred.

201 A write error occurred.

202 Attribute error in the file. PowerBSORT failed in the acquisition of the file attribute.

203 Attribute error in the temporary file. PowerBSORT failed in the acquisition of the file attribute.

204 Error in the file format.

1) Incompatible file formats were specified. Refer to "Supported files type" of the help file.

2) An unsupported file format was specified.

205 The same file was specified for input and output.

206 Error in the record format.

1) A fixed length file and a variable-length file were specified together.

2) The record format of input and output is different.

207 Error in the length of the record.

1) When a variable-length file is specified, this error occurs when the record length of the file attribute exceeds the length specified by MaxRecordLength property.

2) For fixed length files, the length specified by MaxRecordLength property differs from the file attribute record length.

208 The error occurred while opening the file.

209 Too many files are open in the system.

210 The error occurred by closing the file.

211 Hardware media or system software trouble occurred.

212 Insufficient capacity for the temporary file.

213 Sorting was not able to be processed in the memory.

214 The temporary file cannot be generated.

215 There is no file specified for input.

216 No reference permission for the input file.

217 No write permission for the output file.

219 No reference or write permission for the temporary file.

222 Record length was omitted when binary file was specified for input.

224 Write error occurred on the temporary file.

225 Read error occurred on the temporary file.

226 Error in environmental variable setting.

228 Error in initialization file content.

229 No reference permission in the initialization file.

230 Invalid code in the key field.

231 FUJITSU COBOL85 file system error occurred.

1) There is no FUJITSU COBOL85 library.

	2) Error in setting the "Product_Directory" in PowerSORT registry key in Windows NT or Windows 95).
232	The number of symbolic links found while checking the file name exceeded MAXSYMLINKS.
233	The file name is too long.
234	A directory/folder in the file name was not found.
235	The file name specified was a directory.
236	Insufficient space on the output device.
237	The file size exceeded the maximum file size.
238	An error occurred in a system call or library function.
240	The record comparison area was not allocated.
241	A numeric value is not recognized in record summation processing.
243	An error was detected in FUJITSU COBOL85 index file system. Refer to FUJITSU COBOL85 index file error codes for a SubErrorCode property other than 0.
250	Error in the processing of PowerBSORT.
251	Output file already exists.
252	The specified device name is invalid.
253	The input file device name is invalid.
254	The output file device name is invalid.
255	The specified file name is invalid.
256	The input file name is invalid.
257	The output file name is invalid.
258	The specified device cannot be found.
259	The input file device cannot be found.
260	The output file device cannot be found.
261	The file name is too long.
262	The input file name is too long.
263	The output file name is too long.
264	The directory/folder or the file cannot be made.
265	The input file or directory/folder cannot be made.
266	The output file or directory/folder cannot be made.
267	The device is not connected.
268	The input file device is not connected.
269	The output file device is not connected.
270	Directory/Folder is not a sub directory or a root directory.
271	Input file directory/folder is not a sub directory or a root directory.
272	Output file directory/folder is not a sub directory or a root directory.

273 Directory/Folder name is invalid.

274 Input file directory/folder name is invalid.

275 Output file directory/folder name is invalid.

276 Disk unit has failed.

277 Insufficient space on disk.

278 Disk operation failed during access and during retry.

279 Disk recalibrate operation failed during access, and during retry.

280 The reset operation for disk controller is required during hard disk, and reset operation also failed.

281 Disk is in use, or locked.

282 Input file disk is in use, or locked.

283 Output file disk is in use, or locked.

284 Extended error occurred.

285 File or directory/folder is corrupt and unreadable.

286 Opened file is not available. The volume was removed.

287 Opened input file is not available. The volume was removed.

288 Opened output file is not available. The volume was removed.

289 The specified file is not found.

290 The specified input file is not found.

291 The specified output file is not found.

292 File name or extension is too long.

293 Input file name or extension is too long.

294 Output file name or extension is too long.

295 Disk full.

296 Specified drive is not found.

297 Specified input file drive is not found.

298 Specified output file drive is not found.

299 The syntax of the file name, directory/folder name or volume label is invalid.

300 The syntax of the input file name, directory/folder name or volume label is invalid.

301 The syntax of the output file name, directory/folder name or volume label is invalid.

302 An I/O request was not executed by I/O device error.

303 Cannot access the file due to record locks in place.

304 Cannot access the input file due to record locks in place.

305 Cannot access the output file due to record locks in place.

306 File pointer was moved to a position above the top of file.

307 A writing violation occurred on the network.

308 Network access was denied.
309 Network access of input file was denied.
310 Network access of output file was denied.
311 Network is busy.
312 File does not exist after previous access.
313 Network does not exist or is not started.
314 Network of input file does not exist or is not started.
315 Network of output file does not exist or is not started.
316 Specified alias does not exist.
317 Specified alias of input file does not exist.
318 Specified alias of output file does not exist.
319 The volume label of the disk does not exist.
320 The volume label of the input file disk does not exist.
321 The volume label of output file disk does not exist.
322 Network file connection does not exist.
323 Network input file connection does not exist.
324 Network output file connection does not exist.
325 Cannot access specified disk.
326 Cannot access input file disk.
327 Cannot access output file disk.
328 Drive is not ready.
329 Input file drive is not ready.
330 Output file drive is not ready.
331 Network request not supported.
332 Input file network request not supported.
333 Output file network request not supported.
334 Specified device or file cannot be opened.
335 Specified input device or file cannot be opened.
336 Specified output device or file cannot be opened.
337 Specified path is not available at this time.
338 Specified input path is not available at this time.
339 Specified output path is not available at this time.
340 Specified path not found.
341 Specified input path not found.
342 Specified output path not found.
343 Client does not have required privilege.
344 Client does not have required input file privilege.

345 Client does not have required output file privilege.
346 System cannot read from specified device.
347 Remote computer is not available.
348 Network request was not accepted.
349 Drive cannot find required sector.
350 Drive cannot find required input sector.
351 Drive cannot find required output sector.
352 Drive cannot determine specified track or sector.
353 Drive cannot determine specified track or sector for input.
354 Drive cannot determine specified track or sector for output.
355 File pointer cannot be set to device or file.
356 File pointer cannot set to input device or file.
357 File pointer cannot set to output device or file.
358 Remote server is paused in starting.
359 File cannot be accessed. It is in use.
360 Input file cannot be accessed. It is in use.
361 Output file cannot be accessed. It is in use.
362 Input file cannot be opened.
363 Output file cannot be opened.
364 Disk is not recognized. May not be formatted.
365 Input disk is not recognized. May not be formatted.
366 Output disk is not recognized. May not be formatted.
367 Volume does not have a recognized file system. Confirm the file system and volume status.
368 Specified device cannot be written.
369 The device is write protected.
370 Access to the file was denied.
371 Access to the input file was denied.
372 Access to the output file was denied.

Error detail codes (ErrorCode = -3)

See Also

Values of ErrorDetail property when ErrorCode property is -3:

Code	Explanation(property name)
1	Memory necessary for PowerBSORT OLE Control could not be allocated.
2	Error in the specification of UsableMemorySize.(UsableMemorySize)
3	The directory/folder specified was a file name.(TempDir)
4	Invalid number was specified in DisposalNumber property.(DisposalNumber)
5	Specify the input file(s) name.(InputFiles)
6	Invalid number was specified in InputFileType property.(InputFileType)
7	Specify the output file name.(OutputFile)
8	Invalid number was specified in OutputFileType property.(OutputFileType)
9	The specification in KeyCmdStr property is too long.(KeyCmdStr)
10	The self-defined value specified by the record reconstruction feature is not available in the specified length (RconCmdStr)
11	The mistake is found in FUJITSU COBOL85 index information.(FjcobPrimKey , FjcobAlternateKey)
12	The mistake is found in the combination of the equal key specification and the summation feature specification.(HandlingSameKey,SumCmdStr)
13	Invalid number was specified in HandlingSameKey property.(HandlingSameKey)
14	Invalid number was specified in CollationOrder property.(CollationOrder)
15	Invalid number was specified in FieldDefinition property.(FieldDefinition)
16	Invalid number was specified in LineDelimiter property.(LineDelimiter)
17	The FieldDelimiter specification property is too long.(FieldDelimiter)
18	Error in the FieldDelimiter property specification.(FieldDelimiter)
19	The EnableOverwriteInputFile specification is available only with SORT feature.
20	Error in the TempDir property specification(TempDir)
999	An illegal code was detected in the PowerBSORT OCX return value.

Error detail codes(ErrorCode = -4)

See Also

ErrorCode property is -4 is PowerSORT OLE custom control error. Please report **ErrorDetail** property value to our development department.

Error codes for FUJITSU COBOL85 indexed files

See Also

Error codes are from 1 to 4 bytes. The last byte is a file status. The leading byte(s) is(are) a detail code. All values are hexadecimal numbers. A value expressed with a hyphen ("-") indicates that part of the number can be varied through the indicated range.

Example: "30-3E 39" means 3039, 3139, 3239, 3339 ... 3C39, 3D39 or 3E39.

Code	Explanation
00	Normal end.
10	Attempt to read record prior to the first record, or after the last record.
21	Error occurred in the record primary key sequence.
22	A duplicate key occurred when no duplicate was specified.
23	The specified record does not exist.
24	Insufficient disk space.
1 24	Invalid record number(0) while writing to a relative record file.
2 24	An attempt to write too many records to the file.
mmllss 30	Severe error. The ss indicates the originator of the error as detailed below. ll is the low order byte of the error. mm is the high order byte.
mmll00 30	Severe error. The source is uncertain.
mmll01 30	FUJITSU COBOL85 file manager detected operating system error.
101 30	The area allocation failed.
mmll02 30	MS-DOS source
mmll03 30	OS/2 source
mmll04 30	UNIX source
mmll05 30	RM/COS source
mmll06 30	Btrieve source
mmll07 30	RM+DB for INFORMIX source
mmll08 30	RM+DB for ORACLE source
mmll09 30	AmigaDOS
A 30	The error occurred in FUJITSU COBOL85 OpenFileManager.
10A 30	The error occurred in the operating system version.
20A 30	The error occurred in the interface.
30A 30	There is no record.
40A 30	Too many files opened.
50A 30	The error occurred in the handle.
35	The file does not exist.
37	There was a no corresponding access to the file attribute.
7 37	File opened for write access denied. Only Read access is permitted.

38	The file is locked and cannot be opened.
39	The file cannot be opened because the file attribute is incorrect.
1 39	The file organization is incorrect.
2 39	The length of a minimum record is incorrect.
3 39	The length of the maximum record is incorrect.
4 39	The length of a minimum block is incorrect.
5 39	The length of the maximum block is incorrect.
6 39	The delimitation of the record is incorrect.
7 39	The code set is incorrect.
8 39	The collating sequence is incorrect.
9 39	The record format is incorrect.
A 39	The padding character is incorrect.
30-3E 39	The key flag of key 0-E is incorrect.
3F 39	The key flag of key F-FF is incorrect.
40-4E 39	The key offset of key 0-E is incorrect.
4F 39	The key offset of key F-FF is incorrect.
50-5E 39	The key length of key 0-E is incorrect.
5F 39	The key length of key F-FF is incorrect.
43	An error occurred in the last read.
1 90	Illegal access at open processing.
4 90	Access contrary to file organization.
5 90	The file truncate instruction contradicts other users.
6 90	The server session was inhibited.
7 90	Invalid access for a read only file.
8 90	Illegal access.
9 90	The message area is too small.
10 90	Connection Endpoint Identifier is invalid.
92	The file is not closed.
1 92	Open was issued to a file already open.
93	The file is not effective.
2 93	It is not possible to open because of the file lock.
6 93	File has already existed, it is not possible to open.
94	It is an invalid open.
21 94	The file organization is invalid or outside of support.
2 94	The length of a minimum record is invalid.
3 94	The length of the maximum record is invalid.
4 94	The length of a minimum block is invalid.

5 94	The length of the maximum block is invalid.
6 94	The delimitation of the record is invalid.
7 94	The code set is invalid.
8 94	The collating sequence is invalid.
9 94	The record format is invalid.
A 94	The padding character is invalid.
30-3E 94	Key flag of key 0-E is invalid.
3F 94	Key flag of key F-FF is invalid.
40-4E 94	Key offset of key 0-E is invalid.
4F 94	Key offset of key F-FF is invalid.
50-5E 94	The length of key 0-E is invalid.
5F 94	The length of key F-FF is invalid.
60 94	Insufficient memory to open the file.
61 94	Insufficient disk space to open the file.
63 94	The open with lock issued but not supported.
96	The file position is undefined.
1 97	Incorrect character in a line sequential file record.
2 97	Incorrect character.
3 97	Shorter record area than the minimum record length is specified.
4 97	Longer record area than the maximum record length is specified.
7 97	The length of the record is incorrect.
98	File structure invalid.
99	Record is locked by other programs.

Memory Shortages

If memory shortages occur while executing PowerBSORT, do the following:

- Increase the value of **UsableMemorySize** property.
- Cancel other application programs.
- Decrease the number of files.

Temporary File Directories

The following notes specify the directory/folder where temporary files are made in **TempDir** property.

- Make sure the drive and directory/folder exists.
- Make sure the drive has enough empty space.
- Add a backslash(\) to the end of directory/folder name. Errors may occur if it is not added.
- Use a semicolon(;) and no blanks to separate two or more directory names.

[Correct example] C:\;D:\TEMP\

[Incorrect example] C:\ ; D:TEMP

Record Summation

- Summation fields must be defined completely within the record.
- Summation field must not overlap a key field or other summation field.
- The contents of the output record, other than key fields and the summation fields, is unpredictable.
- When an overflow occurs in summary processing, record summation is halted.
- Numbers containing a decimal point cannot be added.

Existing Output Files

When an existing file on a network is specified as the output file, the output file size may not match the actual data size. This can happen because the file size does not change when the amount of data written is smaller than the original file size. Also, the output file size will be **4096** bytes when the input file is a text file and no records are selected for output by the record selection feature. In these cases, the **EOF**(End of File) mark is set correctly with no problem in processing. However, it is recommended to delete existing output files accessed on a network before reusing them with PowerBSORT.

Field Specification with Record Reconstruction

PowerBSORT has Key field, Selection field, Reconstruction field, and Summation field that correspond with Main features and Record option features. Attention is necessary when specifying key fields and summation fields when the record is changed with record reconstruction.

Internal Processing Order of PowerBSORT

PowerBSORT executes processing in the following order:

Data input -> Record selection -> Record reconstruction -> Sort/Merge/Copy -> Record summation -> Data output.

When the record position changes during record reconstruction, the key position for sort/merge processing and record summation processing may be different from the input record. Refer to the following rules to specify each field.

- Selection field

  Selection fields are always specified based on the input record.

- Reconstruction field

  Reconstruction field are always specified based on the input record.

- Key field

  Key fields are usually specified based on the input record. However, they are specifically based on the reconstructed record when record reconstruction is used.

- Summation field

  Summation fields are usually specified based on the input record. However, they are specifically based on the reconstructed record when record reconstruction is used.

SORT(Binary file)

The following sample programs show how to use the SORT feature of PowerBSORT by binary file.

These programs sort external decimal numbers, the first 10 bytes of the existing binary file record "c:\sortin" whose record length is 100 bytes, in ascending order regarding them as a key field . The result is outputted to binary file "c:\sortout".

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False          'Don't display error messages.
    PowerBSORT1.DisposalNumber = 0           'Specify SORT processing.
    PowerBSORT1.InputFiles = "c:\sortin"     'Specify an input file(or two or more
                                             input files).
    PowerBSORT1.InputFileType = 1            'Specify Binary fixed length to input file
                                             format.
    PowerBSORT1.OutputFile = "c:\sortout"    'Specify an output file.
    PowerBSORT1.OutputFileType = 1           'Specify Binary fixed length to output file
                                             format.
    PowerBSORT1.KeyCmdStr = "0.10zdla"       'Specify to sort external decimal number,
                                             the first 10 bytes of record, in ascending
                                             order as a key field.
    PowerBSORT1.MaxRecordLength = 100        'Record length is 100 bytes.
    PowerBSORT1.Action                       'Call PowerBSORT and execute SORT
                                             processing.
    if PowerBSORT1.ErrorCode <> 0 Then       'Check the error code.
     Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
     PowerBSORT1.ErrorDetail
     Exit Sub
    End If
End Sub
```

SORT(Text file)

The following sample programs show how to use the SORT feature of PowerBSORT by text file.

These programs sort ASCII, from byte 20 to byte 30 of the existing text file record "c:\sortin.txt" whose maximum record length is 120 bytes, in ascending order regarding them as a key field. The result is outputted to text file "c:\sortout.txt".

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False              'Don't display error messages.
    PowerBSORT1.DisposalNumber = 0               'Specify SORT processing.
    PowerBSORT1.InputFiles = "c:\sortin.txt"     'Specify an input file(or two or more
                                                 input files).
    PowerBSORT1.InputFileType = 0                'Specify Text to input file format.
    PowerBSORT1.OutputFile = "c:\sortout.txt"    'Specify an output file.
    PowerBSORT1.OutputFileType = 0               'Specify Text to output file format.
    PowerBSORT1.KeyCmdStr = "20.10asca"          'Specify to sort ASCII code, from byte 20
                                                 to byte 30 of input record, in ascending
                                                 order as a key field.
    PowerBSORT1.MaxRecordLength = 120            'Maximum record length is 120 bytes.
    PowerBSORT1.Action                           'Call PowerBSORT and execute SORT
                                                 processing.
    if PowerBSORT1.ErrorCode <> 0 Then           'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

MERGE(Binary file)

The following sample programs show how to use the MERGE feature of PowerBSORT by binary file.

These programs merge two binary files "c:\mrgein1" and "c:\mrgein2" whose record lengths are 100 bytes and are already sorted in ascending order regarding external decimal number, the first 10 bytes of record, as a key field and output the result to binary file "c:\mrgeout"

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False                    'Don't display error messages.
    PowerBSORT1.DisposalNumber = 0                     'Specify MERGE processing.
    PowerBSORT1.InputFiles = "c:\mrgein1 c:\mrgein2"   'Specify some input files.
    PowerBSORT1.InputFileType = 1                      'Specify Binary fixed length to
                                                       input file format.
    PowerBSORT1.OutputFile = "c:\mrgeout"              'Specify an output file.
    PowerBSORT1.OutputFileType = 1                     'Specify Binary fixed length to
                                                       output file format.
    PowerBSORT1.KeyCmdStr = "0.10zdla"                 'External decimal number, the first
                                                       10 bytes of the input record, is
                                                       sorted in ascending order as a key
                                                       field.
    PowerBSORT1.MaxRecordLength = 100                  'Record length is 100 bytes.
    PowerBSORT1.Action                                 'Call PowerBSORT and execute
                                                       MERGE processing.
    if PowerBSORT1.ErrorCode <> 0 Then                 'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

MERGE(Text file)

The following sample programs show how to use the MERGE feature of PowerBSORT by text file.

These programs merge two binary files "c:\mrgein1.txt" and "c:\mrgein2.txt" whose record lengths are 120 bytes and are already sorted in ascending order regarding ASCII, from byte 20 to byte 30 of the record, as a key field and output the result to text file "c:\mrgeout.txt"

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False              'Don't display error messages.
    PowerBSORT1.DisposalNumber = 0               'Specify MERGE processing.
                                                 'Specify some input files.
    PowerBSORT1.InputFiles = "c:\mrgein1.txt c:\mrgein2.txt"
    PowerBSORT1.InputFileType = 0                'Specify Text to input file format.
    PowerBSORT1.OutputFile = "c:\mrgeout.txt"    'Specify an output file.
    PowerBSORT1.OutputFileType = 0               'Specify Text to output file format.
    PowerBSORT1.KeyCmdStr = "20.10asca"          'ASCII code, from byte 10 to byte 20 of
                                                 input record, are already sorted in
                                                 ascending order as a keyfield.
    PowerBSORT1.MaxRecordLength = 120            'Maximum record length is 120 bytes.
    PowerBSORT1.Action                           'Call PowerBSORT and execute MERGE
                                                 processing.
    if PowerBSORT1.ErrorCode <> 0 Then           'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

COPY(Binary file)

The following sample programs show how to use the COPY feature of PowerBSORT by binary file.

These programs output the existing binary file "c:\copyin" whose record length is 100 bytes to a binary file "c:\copyout".

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False          'Don't display error messages.
    PowerBSORT1.DisposalNumber = 2           'Specify COPY processing.
    PowerBSORT1.InputFiles = "c:\copyin"     'Specify an input file(or two or more input
                                             files).
    PowerBSORT1.InputFileType = 1            'Specify Binary fixed length to input file
                                             format.
    PowerBSORT1.OutputFile = "c:\copyout"    'Specify an output file.
    PowerBSORT1.OutputFileType = 1           'Specify Binary fixed length to output file
                                             format.
    PowerBSORT1.MaxRecordLength = 100        'Record length is 100 bytes.
    PowerBSORT1.Action                       'Call PowerBSORT and execute COPY
                                             processing.
    if PowerBSORT1.ErrorCode <> 0 Then       'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

COPY(Text file)

The following sample programs show how to use the COPY feature of PowerBSORT by text file.

These programs output the existing text file "c:\copyin.txt" whose maximum record length is 120 bytes to a text file "c:\copyout.txt".

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False             'Don't display error messages.
    PowerBSORT1.DisposalNumber = 2              'Specify COPY processing.
    PowerBSORT1.InputFiles = "c:\copyin.txt"    'Specify an input file(or two or more input
                                                files).
    PowerBSORT1.InputFileType = 0               'Specify Text to input file format.
    PowerBSORT1.OutputFile = "c:\copyout.txt"   'Specify an output file.
    PowerBSORT1.OutputFileType = 0              'Specify Text to output file format.
    PowerBSORT1.MaxRecordLength = 120           'Maximum record length is 120 bytes.
    PowerBSORT1.Action                          'Call PowerBSORT and execute COPY
                                                processing.
    if PowerBSORT1.ErrorCode <> 0 Then          'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

Using Record Selection feature

The following sample programs show how to use the Record Selection feature of PowerBSORT by binary file.

These programs select the existing binary file "c:\sortin" whose record length is 100 bytes only when byte 10 to byte 14 of the record is bigger than from byte 30 to byte 34, and output to binary file "c:\sortout".

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False          'Don't display error messages.
                                             'Specify to select the record, from byte 10
                                             to byte 14 of it is bigger than from byte 30
                                             to byte 34, as the selection field.
    PowerBSORT1.SelCmdStr = "10.4asc.gt.30.4asc"
    PowerBSORT1.InputFiles = "c:\sortin"     'Specify an input file(or two or more input
                                             files).
    PowerBSORT1.InputFileType = 1            'Specify Binary fixed length to input file
                                             format.
    PowerBSORT1.OutputFile = "c:\sortout"    'Specify an output file.
    PowerBSORT1.OutputFileType = 1           'Specify Binary fixed length to output file
                                             format.
    PowerBSORT1.MaxRecordLength = 100        'Maximum record length is 100 bytes.
    PowerBSORT1.Action                       'Execute Record Selection processing.
    if PowerBSORT1.ErrorCode <> 0 Then       'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

Using Record Reconstruction feature

The following sample programs show how to use the Record Reconstruction feature of PowerBSORT by binary file. These programs reconstruct the record of the existing binary file "c:\sortin" whose record length is 100 bytes in the following order.

from byte 20 to byte 30 -> from byte 30 to byte 40 -> from byte 0 to byte 10

-> from byte 40 to byte 50

The result is outputted to binary file "c:\sortout".

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False              'Don't display error messages.
    PowerBSORT1.RconCmdStr = "20.10 30.10 0.10 40.10"
                                                 'Specify to sort the record in the following
                                                 order as the reconstruction field.   from
                                                 byte 20 to byte 30 -> from byte 30 to byte
                                                 40 -> from byte 0 to byte 10 -> from byte
                                                 40 to byte 50
    PowerBSORT1.InputFiles = "c:\sortin"         'Specify an input file(or two or more input
                                                 files).
    PowerBSORT1.InputFileType = 1                'Specify Binary fixed length to input file
                                                 format.
    PowerBSORT1.OutputFile = "c:\sortout"        'Specify an output file.
    PowerBSORT1.OutputFileType = 1               'Specify Binary fixed length to output file
                                                 format.
    PowerBSORT1.MaxRecordLength = 100            'Maximum record length is 100 bytes.
    PowerBSORT1.Action                           'Execute Record Reconstruction
                                                 processing.
    if PowerBSORT1.ErrorCode <> 0 Then           'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

Using Record Summation feature

The following sample programs show how to use the Record Summation feature of PowerBSORT by text file. These programs sort ASCII, from byte 9 to byte 13 of the existing text file record "c:\sortin.txt" whose record length is 15 bytes, in ascending order regarding them as a key field. Some records having the same value key fields, ASCII, from byte 4 to byte 8, are summarized and the result outputted to text file "c:\sortout".

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False              'Don't display error messages.
    PowerBSORT1.FieldDefinition = 1              'Specify the field with column(byte).
    PowerBSORT1.KeyCmdStr = "9.4asca"            'Specify to sort ASCII, from byte 9 to
                                                 byte 13 of the input record, in ascending
                                                 order as a key field.
    PowerBSORT1.SumCmdStr = "4.4asc"             'Specify the field from byte 4 to byte 8 of
                                                 the input records as a summation key
                                                 field which format is ASCII.
    PowerBSORT1.HandlingSameKey = 3              'When the equal key(the field value of
                                                 key field is equal) exist in the input
                                                 records, sum the value of summation
                                                 field of the input records.
    PowerBSORT1.InputFiles = "c:\sortin.txt"     'Specify an input file(or two or more
                                                 input files).
    PowerBSORT1.InputFileType = 0                'Specify Text to input file format.
    PowerBSORT1.OutputFile = "c:\sortout.txt"    'Specify an output file.
    PowerBSORT1.OutputFileType = 0               'Specify Text to output file format.
    PowerBSORT1.MaxRecordLength = 15             'Maximum record length is 15 bytes.
    PowerBSORT1.Action                           'Execute Record Summation processing.
    if PowerBSORT1.ErrorCode <> 0 Then           'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

Combining Record Selection, Record Reconstruction, and Record Summation feature

The following sample programs show how to combine record processing features of PowerBSORT by text file. These programs select one feature among record selection, record reconstruction and record summation and execute the selected feature with the existing text file

"c:\sortin.txt" whose record length is 15 bytes. The result is outputted to text file "c:\sortout.txt".

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False                 'Don't display error messages.
    PowerBSORT1.FieldDefinition = 1                 'Specify the field with column(byte).
    If Option1.Value = True Then                    'When record selection was checked.
      PowerBSORT1.SelCmdStr = "0.3asc.ne.'DDD'"     'Specify to select record whose first
                                                    3 bytes are not string 'DDD' as a key
                                                    field.
    End If
    If Option2.Value = True Then                    'When record reconstruction was
                                                    checked.
      PowerBSORT1.RconCmdStr = "9.4 3.5 0.3"        'Reconstruct the input record in the
                                                    following order as the reconstruction
                                                    field.   from byte 9 to byte 13 -> from
                                                    byte 3 to byte 8 -> from byte 0 to byte 3
    If Option3.Value = True Then                    'When record summation was checked.
      PowerBSORT1.KeyCmdStr = "9.4asca"             'Reconstruct the input record in the
                                                    following order as a key field. from byte
                                                    4 to byte 9 -> from byte 0 to byte 4 ->
                                                    from byte 9 to byte 13
      PowerBSORT1.SumCmdStr = "4.4asc"              'Specify to summarize ASCII of input
                                                    record, from byte 4 to byte 8, as the
                                                    summation field
      PowerBSORT1.HandlingSameKey = 3               'When the equal key(the field value of
                                                    key field is equal) exist in the input
                                                    records, sum the value of summation
                                                    field of the input records.
    End If
    PowerBSORT1.InputFiles = "c:\sortin.txt"        'Specify an input file(or two or more
                                                    input files).
    PowerBSORT1.InputFileType = 0                   'Specify Text to input file format.
    PowerBSORT1.OutputFile = "c:\sortout.txt"       'Specify an output file.
    PowerBSORT1.OutputFileType = 0                  'Specify Text to input file format.
```

```
    PowerBSORT1.MaxRecordLength = 15          'Maximum record length is 15 bytes.
    PowerBSORT1.Action                        'Execute PowerBSORT processing.
    if PowerBSORT1.ErrorCode <> 0 Then        'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

Using Suppression feature

This sample program shows how to use the Suppression feature of PowerBSORT by binary file.

These programs sort external decimal numbers, from byte 20 to byte 24 of the existing binary file record "c:\sortin" whose record length is 100 bytes, in ascending order regarding them as a key field. Some records having the same value key fields are deleted retaining only one record and the result is outputted to binary file "c:\sortout".

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False          'Don't display error messages.
    PowerBSORT1.DisposalNumber = 0           'Specify SORT processing.
    PowerBSORT1.KeyCmdStr = "20.4zdla"       'Specify to sort external decimal number,
                                             from byte 20 to byte 24 of the input
                                             record, in ascending order as a key field.
    PowerBSORT1.HandlingSameKey = 2          ' When the equal key(the field value of key
                                             field is equal) exist in the input records,
                                             leave only one record and delete other
                                             records.
    PowerBSORT1.InputFiles = "c:\sortin"     'Specify an input file(or two or more input
                                             files).
    PowerBSORT1.InputFileType = 1            'Specify Binary fixed length to input file
                                             format.
    PowerBSORT1.OutputFile = "c:\sortout"    'Specify an output file.
    PowerBSORT1.OutputFileType = 1           'Specify Binary fixed length to output file
                                             format.
    PowerBSORT1.MaxRecordLength = 100        'Record length is 100 bytes.
    PowerBSORT1.Action                       'Execute SORT processing.
    if PowerBSORT1.ErrorCode <> 0 Then       'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

Using FIFO feature

The following sample programs show how to use FIFO feature of PowerBSORT by binary file. These programs sort external decimal numbers, from byte 20 to byte 24 of the existing binary file record "c:\sortin" whose record length is 100 bytes, in ascending order regarding them as a key field. Some records having the same value key fields and inputted originally are outputted first and the result is outputted to binary file "c:\sortout".

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False          'Don't display error messages.
    PowerBSORT1.DisposalNumber = 0           'Specify SORT processing.
    PowerBSORT1.KeyCmdStr = "20.4zdla"       'Specify to sort external decimal number,
                                             from byte 20 to byte 24 of the input
                                             record, in ascending order as a key field.
    PowerBSORT1.HandlingSameKey = 1          'When the equal key(the field value of key
                                             field is equal) exist in the input records,
                                             the first input record encountered with
                                             each unique key value is output first, the
                                             second input record with that same value
                                             is output second, etc.
    PowerBSORT1.InputFiles = "c:\sortin"     'Specify an input file(or two or more input
                                             files).
    PowerBSORT1.InputFileType = 1            'Specify Binary fixed length to input file
                                             format.
    PowerBSORT1.OutputFile = "c:\sortout"    'Specify an output file.
    PowerBSORT1.OutputFileType = 1           'Specify Binary fixed length to input file
                                             format.
    PowerBSORT1.MaxRecordLength = 100        'Record length is 100 bytes.
    PowerBSORT1.Action                       'Execute SORT processing.
    if PowerBSORT1.ErrorCode <> 0 Then       'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

Sorting FUJITSU COBOL85 indexed file

The following sample programs show how to sort FUJITSU COBOL85 fixed length indexed files.

These programs sort ASCII, from byte 5 to byte 8 of the existing FUJITSU COBOL85 fixed length indexed file record "c:\sortcob" whose length is 20 bytes, in ascending order regarding them as a key field. The result is outputted to text file "c:\sortout".

```
Private Sub Command1_Click()
    PowerBSORT1.DispMessage = False          'Don't display error messages.
    PowerBSORT1.DisposalNumber = 0           'Specify SORT processing.
    PowerBSORT1.InputFiles = "c:\sortcob"    'Specify an input file(or two or more input
                                             files).
    PowerBSORT1.InputFileType = 4            'Specify FUJITSU COBOL85 fixed length
                                             indexed file to input file format.
    PowerBSORT1.OutputFile = "c:\sortout"    'Specify an output file.
    PowerBSORT1.OutputFileType = 3           'Specify FUJITSU COBOL85 indexed file to
                                             output file format.
    PowerBSORT1.KeyCmdStr = "5.3asca"        'Specify to sort ASCII code, from byte 5 to
                                             byte 8 of record, in ascending order as a
                                             key field.
    PowerBSORT1.FjcobPrimeKey = "D(5,3)"     'Specify the field from byte 5 to byte 8 of
                                             the input record as the main index key of
                                             FUJITSU COBOL85 indexed file.
    PowerBSORT1.MaxRecordLength = 20         'Maximum record length is 20 bytes.
    PowerBSORT1.Action                       'Execute SORT processing.
    if PowerBSORT1.ErrorCode <> 0 Then       'Check the error code.
      Msgbox "PowerBSORT error was detected." & " ErrorDetail=" &
      PowerBSORT1.ErrorDetail
      Exit Sub
    End If
End Sub
```

See Also

DisposalNumber Property

See Also

SelCmdStr Property

RconCmdStr Property

SumCmdStr Property

HandlingSameKey Property

Record Summation

Field Specification with Record Summation

Internal Processing Order of PowerBSORT

See Also

DisposalNumber Property

SelCmdStr Property

RconCmdStr Property

SumCmdStr Property

HandlingSameKey Property

Record Summation

Field Specification with Record Summation

Internal Processing Order of PowerBSORT

See Also

InputFileType Property

OutputFileType Property

InputFiles Property

OutputFile Property

EnableOverwriteInputFile Property

FjcobAlternateKey Property

FjcobDataCompression Property

FjcobKeyCompression Property

FjcobPrimeKey Property

See Also

CollationOrder Property

CompareAsUpperCase Property

IgnoreControlCode Property

Reverse Property

SkipLeadingBlank Property

See Also

AlphaNumOnly Property

CompareAsUpperCase Property

IgnoreControlCode Property

Reverse Property

SkipLeadingBlank Property

See Also

AlphaNumOnly Property

CollationOrder Property

IgnoreControlCode Property

Reverse Property

SkipLeadingBlank Property

See Also

AlphaNumOnly Property

CollationOrder Property

CompareAsUpperCase Property

Reverse Property

SkipLeadingBlank Property

See Also

AlphaNumOnly Property

CollationOrder Property

CompareAsUpperCase Property

IgnoreControlCode Property

SkipLeadingBlank Property

See Also

AlphaNumOnly Property

CollationOrder Property

CompareAsUpperCase Property

IgnoreControlCode Property

Reverse Property

See Also

ErrorCode Property

ErrorDetail Property

SubErrorCode Property

Error detail codes (ErrorCode = -1 or -2)

Error detail codes (ErrorCode = -3)

Error detail codes (ErrorCode = -4)

Error codes for FUJITSU COBOL85 indexed files

See Also

DispMessage Property

ErrorDetail Property

SubErrorCode Property

Error detail codes (ErrorCode = -1 or -2)

Error detail codes (ErrorCode = -3)

Error detail codes (ErrorCode = -4)

Error codes for FUJITSU COBOL85 indexed files

See Also

DispMessage Property

ErrorCode Property

SubErrorCode Property

Error detail codes (ErrorCode = -1 or -2)

Error detail codes (ErrorCode = -3)

Error detail codes (ErrorCode = -4)

Error codes for FUJITSU COBOL85 indexed files

See Also

DispMessage Property

ErrorCode Property

ErrorDetail Property

Error detail codes (ErrorCode = -1 or -2)

Error detail codes (ErrorCode = -3)

Error detail codes (ErrorCode = -4)

Error codes for FUJITSU COBOL85 indexed files

Key field is shown by position information, the length, the data format, and the order for Sort or Merge processing.

Summation field is used to make one record by the specific field addition on the record with the key field equivalence. Summation field must not overlap a key field or other summation fields. Summation field specifies with no overflow addition.

Selection field specifies some conditions to select specific records during Sort, Merge or Copy processing.

See Also

Introduction to main features

Combination with main feature and record option feature

See Also

InputFiles Property

OutputFile Property

DisposalNumber Property

Reconstruction field is used to change the input record to a new record composition. It is composed of one or more fields of input file record and one or more new constant fields are defined by specifying a Self-defined Value.

See Also

FieldDelimiter Property

SelCmdStr Property

RconCmdStr Property

SumCmdStr Property

HandlingSameKey Property

Record Summation

Field Specification with Record Summation

Internal Processing Order of PowerBSORT

See Also

FieldDefinition Property

LineDelimiter Property

See Also

OutputFileType Property

FjcobDataCompression Property

FjcobKeyCompression Property

FjcobPrimeKey Property

SubErrorCode Property

Error codes for FUJITSU COBOL85 indexed files

See Also

OutputFileType Property

FjcobAlternateKey Property

FjcobKeyCompression Property

FjcobPrimeKey Property

SubErrorCode Property

Error codes for FUJITSU COBOL85 indexed files

See Also

OutputFileType Property

FjcobAlternateKey Property

FjcobDataCompression Property

FjcobPrimeKey Property

SubErrorCode Property

Error codes for FUJITSU COBOL85 indexed files

See Also

OutputFileType Property

FjcobAlternateKey Property

FjcobDataCompression Property

FjcobKeyCompression Property

SubErrorCode Property

Error codes for FUJITSU COBOL85 indexed files

See Also

Introduction to record option features

Combination with main feature and record option feature

KeyCmdStr Property

SumCmdStr Property

See Also

Supported files type

EnableOverwriteInputFile Property

InputFileType Property

MaxRecordLength Property

OutputFile Property

OutputFileType Property

See Also

Supported files type

EnableOverwriteInputFile Property

InputFilesProperty

MaxRecordLength Property

OutputFile Property

OutputFileType Property

See Also

FieldDefinition Property

FieldDelimiter Property

HandlingSameKey Property

LineDelimiter Property

MaxRecordLength Property

RconCmdStr Property

SelCmdStr Property

SumCmdStr Property

See Also

MaxRecordLength Property

See Also

LineDelimiter Property

See Also

Supported files type

EnableOverwriteInputFile Property

InputFilesProperty

InputFileType Property

MaxRecordLength Property

OutputFileType Property

See Also

Supported files type

EnableOverwriteInputFile Property

InputFiles Property

InputFileType Property

MaxRecordLength Property

OutputFile Property

See Also

FieldDefinition Property

FieldDelimiter Property

HandlingSameKey Property

LineDelimiter Property

MaxRecordLength Property

KeyCmdStr Property

SelCmdStr Property

SumCmdStr Property

See Also

FieldDefinition Property

FieldDelimiter Property

HandlingSameKey Property

LineDelimiter Property

MaxRecordLength Property

KeyCmdStr Property

RconCmdStr Property

SumCmdStr Property

See Also

FieldDefinition Property

FieldDelimiter Property

HandlingSameKey Property

LineDelimiter Property

MaxRecordLength Property

KeyCmdStr Property

RconCmdStr Property

SelCmdStr Property

See Also

UsableMemorySize Property

See Also

TempDir Property

See Also

Introduction to main features

Combination with main feature and record option feature

Internal Processing Order of PowerBSORT

DisposalNumber Property

See Also

DispMessage Property

ErrorCode Property

ErrorDetail Property

SubErrorCode Property

Error detail codes (ErrorCode = -3)

Error detail codes (ErrorCode = -4)

Error codes for FUJITSU COBOL85 indexed files

See Also

DispMessage Property

ErrorCode Property

ErrorDetail Property

SubErrorCode Property

Error detail codes (ErrorCode = -1 or -2)

Error detail codes (ErrorCode = -4)

Error codes for FUJITSU COBOL85 indexed files

See Also

DispMessage Property

ErrorCode Property

ErrorDetail Property

SubErrorCode Property

Error detail codes (ErrorCode = -1 or -2)

Error detail codes (ErrorCode = -3)

Error codes for FUJITSU COBOL85 indexed files

See Also

DispMessage Property

ErrorCode Property

ErrorDetail Property

SubErrorCode Property

Error detail codes (ErrorCode = -1 or -2)

Error detail codes (ErrorCode = -3)

Error detail codes (ErrorCode = -4)