System i

Programming
Character Conversion APIs

Version 6 Release 1

System i

Programming
Character Conversion APIs

Version 6 Release 1

Note

Before using this information and the product it supports, read the information in “Notices,” on page 29.

This edition applies to version 6, release 1, modification 0 of IBM i5/OS (product number 5761-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

Contents

Character Conversion APIs

The character conversion APIs are:

- "iconv()—Code Conversion API" (iconv()) converts a buffer of characters from one coded character set identifier (CCSID) into another CCSID.
- "QtqIconvOpen()—Code Conversion Allocation API" on page 9 (QtqIconvOpen()) performs the necessary initializations to convert character encodings and returns a conversion descriptor.
- "iconv_open()—Code Conversion Allocation API" on page 5 (iconv_open()) performs the necessary initializations to convert character encodings and returns a conversion descriptor of type iconv_t.
- "iconv_close()—Code Conversion Deallocation API" on page 12 (iconv_close()) closes the conversion descriptor cd that was initialized by the iconv_open() or QtqIconvOpen() function.
- "Convert Data (QDCXLATE) API" on page 13 (QDCXLATE) converts data through the use of a table object.

The following character conversion APIs are designed to meet the X/Open industry standard functions (formerly Spec 1170).

- **iconv()**
- **iconv_open()**
- **iconv_close()**

Note: The **QtqIconvOpen()** function does not meet the X/Open industry standard.

The three-step conversion provided by the code conversion APIs allows applications to:

- Open a conversion descriptor with a specified CCSID pair (**iconv_open()** or **QtqIconvOpen()** function)
- Do multiple conversions (**iconv()** function)
- Close the conversion descriptor when done (**iconv_close()** function)

This reduces the overhead for applications that need to do multiple conversions using the same CCSID pairs. These functions are performed on the system as entry points in a bindable Integrated Language Environment® (ILE) service program.

Note: The CDRA API, Convert a Graphic Character String (CDRCVRT), converts a graphic character data string from one CCSID to another CCSID.

Top | UNIX-Type APIs | APIs by category

APIs

These are the APIs for this category.

iconv()—Code Conversion API

Syntax

```
#include <iconv.h>

size_t iconv (cd, inbuf, inbytesleft,
              outbuf, outbytesleft)

           iconv_t     cd;
```

```
char        **inbuf;
size_t      *inbytesleft;
char        **outbuf;
size_t      *outbytesleft;
```

Service Program: QTQICONV
Default Public Authority: *USE
Threadsafe: Conditional; see "Usage Notes" on page 5.

The **iconv()** function converts a buffer of characters specified by the *inbuf* parameter from one coded character set identifier (CCSID) into another CCSID and stores the converted characters into a buffer specified by the *outbuf* parameter. (The *inbuf* parameter points to a variable that points to the first character in the input buffer. The *outbuf* parameter points to a variable that points to the first available byte in the output buffer.) The CCSIDs used are those in the conversion descriptor, cd, which was returned from the call to either the **iconv_open()** or the **QtqIconvOpen()** function.

On input, the *inbytesleft* parameter indicates the number of bytes in *inbuf* to be converted. Similarly, the *outbytesleft* parameter indicates the number of bytes available in *outbuf*. These values are decremented when the conversion is done, such that on return they indicate the state of their associated buffers. For encodings dependent on shift state, **iconv()** changes the shift state of the conversion descriptor to match the shift state at the end of the input buffer. For subsequent calls to **iconv()**, conversion begins using the current shift state of the conversion descriptor. The only state-dependent encodings in which **iconv()** supports the updating of the conversion descriptor shift state is mixed-byte EBCDIC.

For encodings dependent on shift state, the conversion descriptor can be returned to its initial shift state by calling **iconv()** with *inbuf* equal to a null pointer, or with *inbuf* pointing to a null pointer. The conversion descriptor can also be set to always return to its initial shift state by specifying the appropriate shift state alternative on the **iconv_open()** and **QtqIconvOpen()** APIs. When **iconv()** is called with the conversion descriptor set in this way, **iconv()** begins conversion from the initial shift state.

If the input buffer ends with an incomplete character or shift sequence, conversion stops after the previous successfully converted bytes. If the output buffer is not large enough to hold the entire converted input, conversion stops just prior to the input bytes that would cause the output buffer to overflow.

During conversion, **iconv()** may encounter valid characters in the input buffer that do not exist in the target CCSID. This is known as a character mismatch. In this case, **iconv()** performs the conversion based on the conversion alternative specified on the **iconv_open()** function.

Authorities and Locks

None.

Parameters

cd INPUT

The conversion descriptor returned by the **iconv_open()** or **QtqIconvOpen()** function that represents the following:

- CCSIDs to convert from and to
- The conversion alternative to use for character mismatches
- The substitution alternative
- The shift-state alternative
- The input length option
- The error option for mixed data

inbuf I/O

A pointer to a variable (pointer) that points to the first character in the input buffer. The variable (pointer) is updated to point to the byte following the last byte successfully used in the conversion. The maximum size of the input buffer is 16 773 104 bytes.

inbytesleft
I/O

A pointer to a variable containing the number of bytes to the end of the input buffer to be converted. This variable is decremented to reflect the number of bytes still not converted in the input buffer. The maximum number of bytes that can be converted is 16 773 104.

outbuf
OUTPUT

A pointer to a variable (pointer) that points to the first available byte in the output buffer. This variable (pointer) is updated to point to the byte following the last byte of converted output data. The maximum size of the output buffer is 16 773 104.

outbytesleft
I/O

A pointer to a variable containing the number of the available bytes to the end of the output buffer. This variable is decremented to reflect the number of bytes still available in the output buffer. The maximum number of bytes available is 16 773 104.

Restrictions

If an error occurs during one of the following requested conversions, the *inbuf*, *inbytesleft*, *outbuf*, and *outbytesleft* parameters may not be updated properly. (ES means encoding scheme.)

- EBCDIC mixed-byte (ES X'1301') to or from ASCII mixed-byte (ES X'2300')
- EBCDIC mixed-byte (ES X'1301') to or from ASCII double-byte (ES X'2200')
- EBCDIC double-byte (ES X'1200') to or from ASCII mixed-byte (ES X'2300')
- EBCDIC double-byte (ES X'1200') to or from ASCII double-byte (ES X'2200')

If the input length option field (on the call to **iconv_open()** or **QtqIconvOpen()**) is set for a NULL-terminated input buffer and the conversion completes successfully, the *inbuf* parameter is not updated for conversions involving the following encoding schemes:

- Encoding scheme X'2300'
- Encoding scheme X'4100'
- Encoding scheme X'4403'
- Encoding scheme X'5100'
- Encoding scheme X'5200'
- Encoding scheme X'5404'
- Encoding scheme X'5405'
- Encoding scheme X'5700'

If the input length option field (on the call to **iconv_open()** or **QtqIconvOpen()**) is set for a NULL-terminated input buffer and an error occurs during the conversion, the *inbytesleft* parameter is not updated for conversions involving the following encoding schemes:

- Encoding scheme X'2300'
- Encoding scheme X'4100'
- Encoding scheme X'4403'
- Encoding scheme X'5100'
- Encoding scheme X'5200'
- Encoding scheme X'5404'

- Encoding scheme X'5405'
- Encoding scheme X'5700'

Return Value

If the entire input buffer is successfully converted, **iconv()** may return the number of nonidentical conversions performed based on the substitution alternative. See "iconv_open()—Code Conversion Allocation API" on page 5 and "QtqIconvOpen()—Code Conversion Allocation API" on page 9. Otherwise, zero will be returned. If an error occurs, **iconv()** returns -1 in the return value, and sets **errno** to indicate the error.

Error Conditions

The following errors can be returned in **errno**:

[E2BIG]
Insufficient space.

Conversion stopped due to lack of space in the output buffer or there was not enough space to store the NULL character in the output buffer.

[EBADDATA]
Shift state not valid in input data

The beginning shift state of the input data buffer does not correspond to the shift state of the conversion descriptor. A shift-state sequence was encountered that tried to change the shift state of the input buffer to the current shift state of the conversion descriptor. For example, an EBCDIC shift-in control character may have been encountered while the conversion descriptor indicated single-byte state. This error is only supported for EBCDIC mixed-byte (X'1301') encoding schemes.

[EBADF]
Descriptor not valid.

The conversion descriptor (cd) parameter is not valid.

[ECONVERT]
The mixed input data contained DBCS characters.

Input conversion stopped due to the occurrence of DBCS characters in the input data when converting from a mixed-byte encoding scheme. The shift state for EBCDIC mixed data remains in the initial single-byte shift state. This error can only be returned when the mixed error option has been set accordingly for the **QtqIconvOpen()** or **iconv_open()** function.

This error is supported only for the following conversions:

- EBCDIC mixed-byte (encoding scheme X'1301') to any single-byte encoding scheme
- ASCII mixed-byte (encoding scheme X'2300') to any single-byte encoding scheme
- EBCDIC mixed-byte (encoding scheme X'1301') to EBCDIC mixed-byte (encoding scheme X'1301')

[EFAULT]
Bad address

The system detected an address that was not valid when attempting to use an argument from the parameter list. An escape message may also be signaled as a result.

[EINVAL]
Parameter not valid.

The conversion stopped because of an incomplete character or shift state sequence at the end of the input buffer.

[ENOBUFS]
Number of bytes for the input or output buffer not valid, or the input length cannot be determined.

The specified number of bytes for *inbytesleft* or *outbytesleft* is not valid. If the input length option field (on the call to **iconv_open()** or **QtqIconvOpen()**) specifies that **iconv()** determines the length of the input buffer and if **iconv()** cannot find a NULL character in the input buffer, this error could be returned.

[ENOMEM]
Not enough space

Insufficient storage space was available to perform the conversion.

[EUNKNOWN]
Undetected error

An undetected error occurred. Contact your service organization. An escape message may also be signaled as a result.

The following escape messages can be signaled:

Message ID	Escape Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF2 E	Error(s) occurred during running of &1 API.

Usage Notes

This API is threadsafe if threads that share a conversion descriptor do not attempt to preserve the shift state.

Related Information

- "iconv_open()—Code Conversion Allocation API"—Code Conversion Allocation API
- "QtqIconvOpen()—Code Conversion Allocation API" on page 9—Code Conversion Allocation API
- "iconv_close()—Code Conversion Deallocation API" on page 12—Code Conversion Deallocation API

API introduced: V3R1

Top | UNIX-Type APIs | APIs by category

---

iconv_open()—Code Conversion Allocation API

Syntax

```
#include <iconv.h>

iconv_t iconv_open (tocode, fromcode)


          char       *tocode;
          char       *fromcode;
```

Service Program: QTQICONV
Default Public Authority: *USE
Threadsafe: Conditional; see "Usage Notes" on page 9.

The **iconv_open()** function performs the necessary initializations to convert character encodings from the source CCSID identified by the *fromcode* parameter to the CCSID identified by the *tocode* parameter. It then returns a conversion descriptor of data type `iconv_t`. For EBCDIC mixed-byte encodings, the conversion descriptor is set to the initial single-byte shift state.

Note: This API performs the same function as the **QtqIconvOpen()** API except that the input types of *fromcode* and *tocode* are character strings.

The conversion descriptor remains valid in a job until that job closes it with **iconv_close()** or the job ends. Keeping unneeded conversion descriptors active results in storage being allocated for these. To reduce this storage, you should call the **iconv_close()** API if you no longer need this conversion descriptor. The number of active descriptors is limited to a maximum of 104 000 active descriptors per process. This number may be reduced if the job has many active threads.

Authorities and Locks

None.

Parameters

tocode INPUT

A pointer to a string containing the CCSID to convert to. See "Format of fromcode and tocode" for the layout of this string.

fromcode

INPUT

A pointer to the string containing the CCSID to convert from, the type of conversion to perform, the substitution alternative, the shift-state alternative, the input length option, and the mixed error option. See "Format of fromcode and tocode" for the layout of this string.

Format of fromcode and tocode

The *fromcode* string consists of the following.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(8)	The word IBMCCSID
8	8	CHAR(5)	Character representation of CCSID number
13	D	CHAR(3)	Conversion alternative
16	10	CHAR(1)	Substitution alternative
17	11	CHAR(1)	Shift-state alternative
18	12	CHAR(1)	Input length option
19	13	CHAR(1)	Error option for mixed data
20	14	CHAR(12)	Reserved

The *tocode* string consists of the following.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	CHAR(8)	The word IBMCCSID
8	8	CHAR(5)	Character representation of CCSID number

Offset		Type	Field
Dec	Hex		
13	D	CHAR(19)	Reserved

Field Descriptions

Character representation of CCSID number. A character representation of the CCSID number to convert from or to. Valid CCSID values are in the range 00001 through 65533. The following special value is supported on the *fromcode* and *tocode* parameters:

00000 The conversion descriptor is created with the CCSID of the current job such that calls to **iconv()** with the conversion descriptor use the CCSID of the job at the time the conversion descriptor was opened by **iconv_open()**. If the CCSID of the current job is 65535 (indicating no conversion), the default CCSID from the DFTCCSID job attribute is used.

Conversion alternative. The conversion alternative that is selected to convert graphic character data. This value is only used on the *fromcode* parameter. The following values can be used:

000 The IBM® -defined default conversion method and the associated conversion tables. Most of the default tables follow the round-trip conversion criterion. For the default tables that do not follow the round-trip conversion criterion, see the i5/OS® globalization topic collection.

057 The enforced subset match (substitution) criterion. For the CCSID conversion pairs that support this criterion, see i5/OS globalization.

102 The best-fit conversion criterion for character mismatch.

Error option for mixed data. Whether **iconv()** returns an error when it converts a character string from a mixed-byte encoding scheme and the input buffer contains double-byte character set (DBCS) characters.

0 An error is not returned when converting from a mixed-byte encoding scheme and the input buffer contains DBCS characters. Instead, **iconv()** converts the DBCS characters to the CDRA-standardized control character substitute of the target CCSID.

1 An ECONVERT error is returned in **errno** when **iconv()** encounters a DBCS character in the input buffer when converting from a mixed-byte encoding scheme.

Input length option. Whether **iconv()** determines the number of bytes to the end of the input buffer being converted. This value is only used on the *fromcode* parameter. The possible values follow:

0 The conversion descriptor is created such that the **iconv()** function does not determine the number of bytes to the end of the input buffer being converted; therefore, a valid value for the *inbytesleft* parameter must be specified.

1 The conversion descriptor is created such that the **iconv()** function determines the number of bytes to the end of the input buffer being converted; therefore, the input buffer must end with a NULL character. The resulting converted output buffer is ended with a NULL character by **iconv()**. The *inbytesleft* parameter must point to a value of zero.

Reserved. A reserved field that must be set to hexadecimal zeros.

Shift-state alternative. Whether the shift state should be restored to its initial shift state before conversion is performed by **iconv()**. This value is only used on the *fromcode* parameter. The following values can be used:

0 The conversion descriptor is not returned to its initial shift state.

Note: The conversion descriptor can still be returned to its initial shift state by making a separate call to **iconv()**. *inbuf* must be equal to a null pointer or pointing to a null pointer.

1 The conversion descriptor is always returned to its initial shift state.

Substitution alternative. Whether the number of substitution characters encountered in the converted output data is returned by **iconv()**. This value is only used on the *fromcode* parameter. The following values can be used:

0 The number of substitution characters encountered is not returned.

1 The number of substitution characters encountered is returned. This number may be equal to or greater than the actual number of nonidentical conversions (substitutions) performed. The substitution character for a particular encoding scheme is defined in the Character Data Representation Architecture (CDRA).

Note: This value for the substitution alternative is valid only when using the enforced subset conversion criterion (057) for the conversion alternative field. Performance is decreased when using this alternative.

The word IBMCCSID. The 8-byte character field that contains the characters I, B, M, C, C, S, I, D. The characters must be in uppercase.

Return Value

If successful, **iconv_open()** returns a conversion descriptor of data type `iconv_t`. This conversion descriptor must be passed unchanged as an input parameter to the **iconv()** and **iconv_close()** functions.

If unsuccessful, **iconv_open()** returns -1 in the return value of the conversion descriptor and sets **errno** to indicate the error.

Error Conditions

The following errors can be returned in **errno**:

[EFAULT]
Bad address

The system detected an address that was not valid when attempting to use an argument from the parameter list. An escape message may also be signaled as a result.

[EINVAL]
Parameter not valid.

The conversion specified in the *fromcode* and *tocode* parameters is not supported.

When an errno value of EINVAL is returned, check the *fromcode* and *tocode* parameters for CCSIDs that are not valid or unsupported alternatives and options.

[ENOMEM]
Not enough space.

Insufficient storage space is available.

[EUNKNOWN]
Undetected error

An undetected error occurred. Contact your service organization. An escape message may also be signaled as a result.

The following escape messages can be signaled:

Message ID	Escape Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF2 E	Error(s) occurred during running of &1 API.

Usage Notes

This API is threadsafe if threads that share a conversion descriptor do not attempt to preserve the shift state.

Related Information

- "iconv()—Code Conversion API" on page 1—Code Conversion API
- "iconv_close()—Code Conversion Deallocation API" on page 12—Code Conversion Deallocation API

API introduced: V3R1

QtqIconvOpen()—Code Conversion Allocation API

```
iconv_t QtqIconvOpen (tocode, fromcode)


            QtqCode_T   *tocode;
            QtqCode_T   *fromcode;
```

Service Program: QTQICONV
Default Public Authority: *USE
Threadsafe: Yes

The **QtqIconvOpen()** function performs the necessary initializations to convert character encodings from the source CCSID identified by the *fromcode* to the CCSID identified by the *tocode*. It then returns a conversion descriptor of data type `iconv_t`. For EBCDIC mixed-byte encodings, the conversion descriptor is set to the initial single-byte shift state.

Note: This API performs the same function as the **iconv_open()** API except that the input type of *fromcode* and *tocode* is of data type `QtqCode_T`.

The conversion descriptor remains valid in a job until that job closes with **iconv_close()** or the job ends. Keeping unneeded conversion descriptors active results in storage being allocated for these. To reduce this storage, you should call the **iconv_close()** API if you no longer need this conversion descriptor. The number of active descriptors is limited to a maximum of 104 000 active descriptors per process. This number may be reduced if the job has many active threads.

Authorities and Locks

None.

Parameters

tocode INPUT

A pointer to the structure containing the CCSID to convert to. For the layout of this structure, see "Format of QtqCode_T" on page 10.

fromcode

INPUT

A pointer to the structure containing the CCSID to convert from, the type of conversion to perform, the substitution alternative, and the shift-state alternative. For the layout of this structure, see "Format of QtqCode_T."

Format of QtqCode_T

The **QtqCode_T** structure consists of the following.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	CCSID
4	4	BINARY(4)	Conversion alternative
8	8	BINARY(4)	Substitution alternative
12	C	BINARY(4)	Shift-state alternative
16	10	BINARY(4)	Input length option
20	14	BINARY(4)	Error option for mixed data
24	18	CHAR(8)	Reserved

Field Descriptions

CCSID. The CCSID to convert from or to. Valid CCSID values are in the range 1 through 65533. The following special value is supported on the *fromcode* and *tocode* parameters:

0 The conversion descriptor is created with the CCSID of the current job such that calls to **iconv()** with the conversion descriptor use the CCSID of the job at the time the conversion descriptor was opened by **iconv_open()**. If the CCSID of the current job is 65535 (indicating no conversion), the default CCSID from the DFTCCSID job attribute is used.

Conversion alternative. The conversion alternative that is selected to convert graphic character data. This value is only used on the *fromcode* parameter; it is ignored on the *tocode* parameter. The following values can be used:

0 The IBM-defined default conversion method and the associated conversion tables. Most of the default tables follow the round-trip conversion criterion. For the default tables that do not follow the round-trip conversion criterion, see the i5/OS® globalization topic collection.

57 The enforced subset match (substitution) criterion. For the CCSID conversion pairs that support this criterion, see the i5/OS globalization topic collection.

102 The best-fit conversion criterion for character mismatch.

Error option for mixed data. Whether **iconv()** returns an error when it converts a character string from a mixed-byte encoding scheme and the input buffer contains double-byte character (DBCS) characters.

Note: This is only valid on the fromcode parameter and only used if the target CCSID is SBCS.

Valid values are:

0 An error is not returned when converting from a mixed-byte encoding scheme and the input buffer contains DBCS characters. Instead, **iconv()** converts the DBCS characters to the CDRA-standardized control character substitute of the target CCSID.

1 An ECONVERT error is returned in **errno** when **iconv()** encounters a DBCS character in the input buffer when converting from a mixed-byte encoding scheme.

Input length option. Whether **iconv()** determines the number of bytes to the end of the input buffer being converted. This value is only used on the *fromcode* parameter. The possible values follow:

0 The conversion descriptor is created such that the **iconv()** function does not determine the number of bytes; therefore, a valid value for the *inbytesleft* parameter must be specified.

1 The conversion descriptor is created such that the **iconv()** function determines the number of bytes to the end of the input buffer to be converted; therefore, the input buffer must end with a NULL character. The resulting converted output buffer is ended with a NULL character by **iconv()**. The *inbytesleft* parameter must point to a value of zero.

Reserved. A reserved field that must be set to hexadecimal zeros.

Shift-state alternative. Whether the shift state should be restored to its initial shift state before conversion is performed by **iconv()**. This value is only used on the *fromcode* parameter; it is ignored on the *tocode* parameter. The following values can be used:

0 The conversion descriptor is not returned to its initial shift state.

Note: The conversion descriptor can still be returned to its initial shift state by making a separate call to **iconv()**. The *inbuf* must be equal to a null pointer or pointing to a null pointer.

1 The conversion descriptor is always returned to its initial shift state.

Substitution alternative. Whether the number of substitution characters encountered in the converted output data is returned by **iconv()**. This value is only used on the *fromcode* parameter; it is ignored on the *tocode* parameter. The following values can be used:

0 The number of substitution characters encountered is not returned.

1 The number of substitution characters encountered is returned. This number may be equal to or greater than the actual number of nonidentical conversions (substitutions) performed. The substitution character for a particular encoding scheme is defined in the Character Data Representation Architecture (CDRA).

Note: This value for the substitution alternative is valid only when using the enforced subset conversion criterion for the conversion alternative. Performance is decreased when using this alternative.

Return Value

If successful, **QtqIconvOpen()** returns a conversion descriptor of data type `iconv_t`. This conversion descriptor must be passed unchanged as an input parameter to the **iconv()** and **iconv_close()** functions.

If unsuccessful, **QtqIconvOpen()** returns -1 and in the return value of the conversion descriptor and sets **errno** to indicate the error.

Error Conditions

The following errors can be returned in **errno**:

[EFAULT]

Bad address

The system detected an address that was not valid when attempting to use an argument from the parameter list. An escape message may also be signaled as a result.

[EINVAL]
Parameter not valid.

The conversion specified in the *fromcode* and *tocode* parameters is not supported.

When an errno value of EINVAL is returned, check the *fromcode* and *tocode* parameters for CCSIDs that are not valid or unsupported alternatives and options.

[ENOMEM]
Not enough space.

Insufficient storage space is available.

[EUNKNOWN]
Undetected error

An undetected error occurred. Contact your service organization. An escape message may also be signaled as a result.

The following escape messages can be signaled:

Message ID	Escape Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF2 E	Error(s) occurred during running of &1 API.

Related Information

- "iconv()—Code Conversion API" on page 1—Code Conversion API
- "iconv_close()—Code Conversion Deallocation API"—Code Conversion Deallocation API

API introduced: V3R1

Top | UNIX-Type APIs | APIs by category

---

iconv_close()—Code Conversion Deallocation API

Syntax

```
#include <iconv.h>

int iconv_close (cd)

          iconv_t     cd;
```

Service Program: QTQICONV
Default Public Authority: *USE
Threadsafe: Conditional; see "Usage Notes" on page 13.

The **iconv_close()** function closes the conversion descriptor cd that was initialized by the **iconv_open()** or **QtqIconvOpen()** function.

Authorities and Locks

None.

Parameters

cd INPUT

The conversion descriptor returned by a previous successful call to **iconv_open()** or **QtqIconvOpen()**.

Return Value

If an error occurs, **iconv_close()** returns a value of -1 and **errno** is set to indicate the error. If **iconv_close()** completes successfully, a value of zero is returned.

Error Conditions

The following errors can be returned in **errno**:

[EBADF]
Descriptor not valid.

The conversion descriptor (cd) parameter is not valid.

[EUNKNOWN]
Undetected error

An undetected error occurred. Contact your service organization. An escape message may also be signaled as a result.

The following escape messages can be signaled:

Message ID	Escape Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF2 E	Error(s) occurred during running of &1 API.

Usage Notes

This API is threadsafe if threads that share a conversion descriptor do not attempt to preserve the shift state.

Related Information

- "iconv()—Code Conversion API" on page 1—Code Conversion API
- "iconv_open()—Code Conversion Allocation API" on page 5—Code Conversion Allocation API
- "QtqIconvOpen()—Code Conversion Allocation API" on page 9—Code Conversion Allocation API

API introduced: V3R1

Top | UNIX-Type APIs | APIs by category

Convert Data (QDCXLATE) API

Required Parameter Group:

1	Length of data being converted	Input	Packed(5,0)
2	Conversion data	I/O	Char(*)
3	SBCS conversion table name	Input	Char(10)

Optional Parameter Group:

4	SBCS conversion table library name	Input	Char(10)
5	Output data	Output	Char(*)
6	Length of output buffer	Input	Packed(5,0)
7	Length of converted data	Output	Packed(5,0)

8	DBCS language	Input	Char(10)
9	Shift-out and shift-in characters	Input	Char(1)
10	Type of conversion	Input	Char(10)

Default Public Authority: *USE
Threadsafe: Yes

The Convert Data (QDCXLATE) API converts data through the use of a table object. (If you need to convert the case of your data, it is recommended that you use the Convert Case (QLGCNVCS, QlgConvertCase API.) You also can use the equivalent API QTBXLATE to achieve the same function. The call interface to QTBXLATE is identical to Convert Data (QDCXLATE).

» This API is available for compatibility purposes or user-defined mappings only. Do not use this API in new development; instead, use the "iconv()—Code Conversion API" on page 1 or the Convert a Graphic Character String (CDRCVRT) API. «

You can create the conversion table that QDCXLATE uses for the conversion, or you can use an IBM® -supplied table. The IBM-supplied tables can be found in the QUSRSYS library. For a list of both the conversion tables and the casing tables, see the i5/OS® globalization topic collection. These tables are not the same as those used by the intersystem communications function (ICF) for conversion support. For more information, see Socket programming.

You can create your own conversion tables using the Create Table (CRTTBL) command.

When the QDCXLATE API is called with parameters 1, 2, 3, and 4, it converts single-byte data. When all parameters are specified, DBCS conversion is taking place.

The QDCXLATE API can distinguish double-byte from single-byte characters when converting from EBCDIC to ASCII and from ASCII to EBCDIC if the proper parameters have been supplied. The QDCXLATE API converts data byte for byte and returns the converted data to your program.

When only single-byte data is converted, the input (unconverted) data is replaced with the converted data. When double-byte data is converted, the converted data is placed in the output data parameter.

The QDCXLATE API is thread safe only when converting single-byte data or T.61 data.

Authorities and Locks

Table Authority
*USE

Table Library Authority
*USE

Required Parameter Group

Length of data being converted
INPUT; PACKED(5,0)

The length of the data being converted. This value cannot exceed 32 767.

Conversion data
I/O; CHAR(*)

The data to be converted. This buffer also contains the output data after conversion when the API is called with only the required parameter group.

SBCS conversion table name
INPUT; CHAR(10)

The name of the single-byte character set (SBCS) conversion table to be used. The table may be a system-supplied or user-supplied conversion table. The table name must be left-justified.

Note: This parameter is ignored when the DBCS language parameter is set to *BG5, *KSC, *SCGS *J90X5026, *J90X5035, or *SCGBK.

Optional Parameter Group

SBCS conversion table library name
INPUT; CHAR(10)

The name of the library that contains the SBCS conversion table. The library name must be left-justified. If this parameter is not specified, the library list is used to locate the conversion table. This parameter is ignored when the DBCS language parameter is set to *BG5, *KSC, *SCGS, *J90X5026, *J90X5035 or *SCGBK.

Output data
OUTPUT; CHAR(*)

The output buffer that contains the double-byte character set (DBCS) data that was converted.

Because of the insertion of shift-out and shift-in characters, it is possible that the converted data is longer than the source data. If this is the case, it is not possible to do the conversion in place, as is done when you use only a required-parameter-group call. The converted data is then placed in the area pointed to by this parameter.

Length of output buffer
INPUT; PACKED(5,0)

The size of the output data buffer. The maximum length should match the actual size of the output data parameter. If the converted output is longer than the length of output buffer parameter, an exception is signaled.

Length of converted data
OUTPUT; PACKED(5,0)

The actual length of the converted output in the output data parameter.

DBCS language
INPUT; CHAR(10)

The DBCS language that is being converted. All values must be padded on the right with blanks. The possible values follow:

**JPN*	IBM Japanese graphic character set
**KOR*	IBM Korean graphic character set
**CHS*	IBM Simplified Chinese graphic character set
**CHT*	IBM Traditional Chinese graphic character set
**BG5*	Taiwan industry standard graphic character set (BIG-5)
**KSC*	Korean industry standard graphic character set (KS)
**SCGS*	The People's Republic of China National standard graphic character set (GB)
**J90X5026*	The Japanese JIS X 0208 1990 standard mapped using CCSID 5026.
**J90X5035*	The Japanese JIS X 0208 1990 standard mapped using CCSID 5035.
**SCGBK*	The People's Republic of China National standard graphic character set extended (GBK)

Shift-out and shift-in characters
INPUT; CHAR(1)

Whether shift-out and shift-in characters should be inserted during the conversion. This parameter is ignored when the DBCS language parameter is set to *BG5, *KSC, *SCGS, *J90X5026, *J90X5035 or *SCGBK. The possible values follow:

Y Insert shift-out and shift-in characters
N Do not insert shift-out and shift-in characters

Type of conversion
INPUT; CHAR(10)

The type of DBCS conversion being done. The possible values follow:

**AE* Convert ASCII to EBCDIC
**EA* Convert EBCDIC to ASCII

Note: You are responsible for specifying the correct SBCS table name for the type of conversion being done by this DBCS request except when the DBCS language parameter is set to *BG5, *KSC, *SCGS, *J90X5026, *J90X5035 or *SCGBK.

Error Messages

Message ID	Error Message Text
CPF2143 E	Cannot allocate object &1 in &2 type *&3.
CPF2144 E	Not authorized to &1 in &2 type *&3.
CPF24B4 E	Severe error while addressing parameter list.
CPF2619 E	Table &1 not found.
CPF264D E	Double-byte character set language not valid.
CPF264E E	Shift-in and shift-out value of double-byte character set not valid.
CPF264F E	Translation type of double-byte character set not valid.
CPF2647 E	Buffer length not valid.
CPF265E E	Number of parameters specified not valid.
CPF265F E	Translation length exceeded maximum length.
CPF2651 E	Table &1 not found.
CPF2669 E	Double-byte character set source not valid.
CPF269A E	Library parameter is not set to “QSYS ” on call.
CPF269B E	T.61 conversion table not found.
CPF269C E	Error in input data.
CPF269D E	A nonspacing underline character was found in the input data.
CPF3C90 E	Literal value cannot be changed.
CPF6309 E	Not authorized to library &1.
CPF9802 E	Not authorized to object &2 in &3.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API existed prior to V1R3

Top | UNIX-Type APIs | APIs by category

Concepts

These are the concepts for this category.

Header Files for UNIX-Type Functions

Programs using the UNIX®-type functions must include one or more header files that contain information needed by the functions, such as:

- Macro definitions
- Data type definitions
- Structure definitions

- Function prototypes

The header files are provided in the QSYSINC library, which is optionally installable. Make sure QSYSINC is on your system before compiling programs that use these header files. For information about installing the QSYSINC library, see Include files and the QSYSINC library.

The table below shows the file and member name in the QSYSINC library for each header file used by the UNIX-type APIs in this publication.

Name of Header File	Name of File in QSYSINC	Name of Member
arpa/inet.h	ARPA	INET
arpa/nameser.h	ARPA	NAMESER
bse.h	H	BSE
bsedos.h	H	BSEDOS
bseerr.h	H	BSEERR
dirent.h	H	DIRENT
errno.h	H	ERRNO
fcntl.h	H	FCNTL
grp.h	H	GRP
inttypes.h	H	INTTYPES
limits.h	H	LIMITS
netdbh.h	H	NETDB
netinet/icmp6.h	NETINET	ICMP6
net/if.h	NET	IF
netinet/in.h	NETINET	IN
netinet/ip_icmp.h	NETINET	IP_ICMP
netinet/ip.h	NETINET	IP
netinet/ip6.h	NETINET	IP6
netinet/tcp.h	NETINET	TCP
netinet/udp.h	NETINET	UDP
netns/idp.h	NETNS	IDP
netns/ipx.h	NETNS	IPX
netns/ns.h	NETNS	NS
netns/sp.h	NETNS	SP
net/route.h	NET	ROUTE
nettel/tel.h	NETTEL	TEL
os2.h	H	OS2
os2def.h	H	OS2DEF
pwd.h	H	PWD
Qlg.h	H	QLG
qp0lchsg.h	H	QP0LCHSG
qp0lflop.h	H	QP0LFLOP
qp0ljrnl.h	H	QP0LJRNL
qp0lror.h	H	QP0LROR

Name of Header File	Name of File in QSYSINC	Name of Member
qp0lrro.h	H	QP0LRRO
qp0lrtsg.h	H	QP0LRTSG
qp0lscan.h	H	QP0LSCAN
Qp0lstdi.h	H	QP0LSTDI
qp0wpid.h	H	QP0WPID
qp0zdipc.h	H	QP0ZDIPC
qp0zipc.h	H	QP0ZIPC
qp0zolip.h	H	QP0ZOLIP
qp0zolsm.h	H	QP0ZOLSM
qp0zripc.h	H	QP0ZRIPC
qp0ztrc.h	H	QP0ZTRC
qp0ztrml.h	H	QP0ZTRML
qp0z1170.h	H	QP0Z1170
qsoasync.h	H	QSOASYNC
qtnxaapi.h	H	QTNXAAPI
qtnxadtp.h	H	QTNXADTP
qtomeapi.h	H	QTOMEAPI
qtossapi.h	H	QTOSSAPI
resolv.h	H	RESOLV
semaphore.h	H	SEMAPHORE
signal.h	H	SIGNAL
spawn.h	H	SPAWN
ssl.h	H	SSL
sys/errno.h	H	ERRNO
sys/ioctl.h	SYS	IOCTL
sys/ipc.h	SYS	IPC
sys/layout.h	SYS	LAYOUT
sys/limits.h	H	LIMITS
sys/mman.h	SYS	MMAN
sys/msg.h	SYS	MSG
sys/param.h	SYS	PARAM
sys/resource.h	SYS	RESOURCE
sys/sem.h	SYS	SEM
sys/setjmp.h	SYS	SETJMP
sys/shm.h	SYS	SHM
sys/signal.h	SYS	SIGNAL
sys/socket.h	SYS	SOCKET
sys/stat.h	SYS	STAT
sys/statvfs.h	SYS	STATVFS
sys/time.h	SYS	TIME
sys/types.h	SYS	TYPES

Name of Header File	Name of File in QSYSINC	Name of Member
sys/uio.h	SYS	UIO
sys/un.h	SYS	UN
sys/wait.h	SYS	WAIT
ulimit.h	H	ULIMIT
unistd.h	H	UNISTD
utime.h	H	UTIME

You can display a header file in QSYSINC by using one of the following methods:

- Using your editor. For example, to display the **unistd.h** header file using the Source Entry Utility editor, enter the following command:

  ```
  STRSEU SRCFILE(QSYSINC/H) SRCMBR(UNISTD) OPTION(5)
  ```

- Using the Display Physical File Member command. For example, to display the **sys/stat.h** header file, enter the following command:

  ```
  DSPPFM FILE(QSYSINC/SYS) MBR(STAT)
  ```

You can print a header file in QSYSINC by using one of the following methods:

- Using your editor. For example, to print the **unistd.h** header file using the Source Entry Utility editor, enter the following command:

  ```
  STRSEU SRCFILE(QSYSINC/H) SRCMBR(UNISTD) OPTION(6)
  ```

- Using the Copy File command. For example, to print the **sys/stat.h** header file, enter the following command:

  ```
  CPYF FROMFILE(QSYSINC/SYS) TOFILE(*PRINT) FROMMBR(STAT)
  ```

Symbolic links to these header files are also provided in directory /QIBM/include.

Top | UNIX-Type APIs | APIs by category

Errno Values for UNIX-Type Functions

Programs using the UNIX®-type functions may receive error information as *errno* values. The possible values returned are listed here in ascending *errno* value sequence.

Name	Value	Text	Details
EDOM	3001	A domain error occurred in a math function.	
ERANGE	3002	A range error occurred.	
ETRUNC	3003	Data was truncated on an input, output, or update operation.	
ENOTOPEN	3004	File is not open.	You attempted to do an operation that required the file to be open.
ENOTREAD	3005	File is not opened for read operations.	You tried to read a file that is not open for read operations.
EIO	3006	Input/output error.	A physical I/O error occurred or a referenced object was damaged.
ENODEV	3007	No such device.	

Name	Value	Text	Details
ERECIO	3008	Cannot get single character for files opened for record I/O.	The file that was specified is open for record I/O and you attempted to read it as a stream file.
ENOTWRITE	3009	File is not opened for write operations.	You tried to update a file that has not been opened for write operations.
ESTDIN	3010	The stdin stream cannot be opened.	
ESTDOUT	3011	The stdout stream cannot be opened.	
ESTDERR	3012	The stderr stream cannot be opened.	
EBADSEEK	3013	The positioning parameter in fseek is not correct.	
EBADNAME	3014	The object name specified is not correct.	
EBADMODE	3015	The type variable specified on the open function is not correct.	The mode that you attempted to open the file in is not correct.
EBADPOS	3017	The position specifier is not correct.	
ENOPOS	3018	There is no record at the specified position.	You attempted to position to a record that does not exist in the file.
ENUMMBRS	3019	Attempted to use ftell on multiple members.	Remove all but one member from the file.
ENUMRECS	3020	The current record position is too long for ftell.	
EINVAL	3021	The value specified for the argument is not correct.	A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.
EBADFUNC	3022	Function parameter in the signal function is not set.	
ENOENT	3025	No such path or directory.	The directory or a component of the path name specified does not exist.
ENOREC	3026	Record is not found.	
EPERM	3027	The operation is not permitted.	You must have appropriate privileges or be the owner of the object or other resource to do the requested operation.
EBADDATA	3028	Message data is not valid.	The message data that was specified for the error text is not correct.
EBUSY	3029	Resource busy.	An attempt was made to use a system resource that is not available at this time.
EBADOPT	3040	Option specified is not valid.	
ENOTUPD	3041	File is not opened for update operations.	
ENOTDLT	3042	File is not opened for delete operations.	
EPAD	3043	The number of characters written is shorter than the expected record length.	The length of the record is longer than the buffer size that was specified. The data written was padded to the length of the record.

Name	Value	Text	Details
EBADKEYLN	3044	A length that was not valid was specified for the key.	You attempted a record I/O against a keyed file. The key length that was specified is not correct.
EPUTANDGET	3080	» A write operation should not immediately follow a read operation. «	
EGETANDPUT	3081	» A read operation should not immediately follow a write operation. «	
EIOERROR	3101	A nonrecoverable I/O error occurred.	
EIORECERR	3102	A recoverable I/O error occurred.	
EACCES	3401	Permission denied.	An attempt was made to access an object in a way forbidden by its object access permissions.
ENOTDIR	3403	Not a directory.	A component of the specified path name existed, but it was not a directory when a directory was expected.
ENOSPC	3404	No space is available.	The requested operations required additional space on the device and there is no space left. This could also be caused by exceeding the user profile storage limit when creating or transferring ownership of an object.
EXDEV	3405	Improper link.	A link to a file on another file system was attempted.
EAGAIN	3406	Operation would have caused the process to be suspended.	
EWOULDBLOCK	3406	Operation would have caused the process to be suspended.	
EINTR	3407	Interrupted function call.	
EFAULT	3408	The address used for an argument was not correct.	In attempting to use an argument in a call, the system detected an address that is not valid.
ETIME	3409	Operation timed out.	
ENXIO	3415	No such device or address.	
ECLOSED	3417	Socket closed.	
EAPAR	3418	Possible APAR condition or hardware failure.	
ERECURSE	3419	Recursive attempt rejected.	
EADDRINUSE	3420	Address already in use.	
EADDRNOTAVAIL	3421	Address is not available.	
EAFNOSUPPORT	3422	The type of socket is not supported in this protocol family.	
EALREADY	3423	Operation is already in progress.	
ECONNABORTED	3424	Connection ended abnormally.	
ECONNREFUSED	3425	A remote host refused an attempted connect operation.	

Name	Value	Text	Details
ECONNRESET	3426	A connection with a remote socket was reset by that socket.	
EDESTADDRREQ	3427	Operation requires destination address.	
EHOSTDOWN	3428	A remote host is not available.	
EHOSTUNREACH	3429	A route to the remote host is not available.	
EINPROGRESS	3430	Operation in progress.	
EISCONN	3431	A connection has already been established.	
EMSGSIZE	3432	Message size is out of range.	
ENETDOWN	3433	The network is currently not available.	
ENETRESET	3434	A socket is connected to a host that is no longer available.	
ENETUNREACH	3435	Cannot reach the destination network.	
ENOBUFS	3436	There is not enough buffer space for the requested operation.	
ENOPROTOOPT	3437	The protocol does not support the specified option.	
ENOTCONN	3438	Requested operation requires a connection.	
ENOTSOCK	3439	The specified descriptor does not reference a socket.	
ENOTSUP	3440	Operation is not supported.	The operation, though supported in general, is not supported for the requested object or the requested arguments.
EOPNOTSUPP	3440	Operation is not supported.	The operation, though supported in general, is not supported for the requested object or the requested arguments.
EPFNOSUPPORT	3441	The socket protocol family is not supported.	
EPROTONOSUPPORT	3442	No protocol of the specified type and domain exists.	
EPROTOTYPE	3443	The socket type or protocols are not compatible.	
ERCVDERR	3444	An error indication was sent by the peer program.	
ESHUTDOWN	3445	Cannot send data after a shutdown.	
ESOCKTNOSUPPORT	3446	The specified socket type is not supported.	
ETIMEDOUT	3447	A remote host did not respond within the timeout period.	

Name	Value	Text	Details
EUNATCH	3448	The protocol required to support the specified address family is not available at this time.	
EBADF	3450	Descriptor is not valid.	A file descriptor argument was out of range, referred to a file that was not open, or a read or write request was made to a file that is not open for that operation.
EMFILE	3452	Too many open files for this process.	An attempt was made to open more files than allowed by the value of OPEN_MAX. The value of OPEN_MAX can be retrieved using the sysconf() function.
ENFILE	3453	Too many open files in the system.	A system limit has been reached for the number of files that are allowed to be concurrently open in the system.
EPIPE	3455	Broken pipe.	
ECANCEL	3456	Operation cancelled.	
EEXIST	3457	Object exists.	The object specified already exists and the specified operation requires that it not exist.
EDEADLK	3459	Resource deadlock avoided.	An attempt was made to lock a system resource that would have resulted in a deadlock situation. The lock was not obtained.
ENOMEM	3460	Storage allocation request failed.	A function needed to allocate storage, but no storage is available.
EOWNERTERM	3462	The synchronization object no longer exists because the owner is no longer running.	The process that had locked the mutex is no longer running, so the mutex was deleted.
EDESTROYED	3463	The synchronization object was destroyed, or the object no longer exists.	
ETERM	3464	Operation was terminated.	
ENOENT1	3465	No such file or directory.	A component of a specified path name did not exist, or the path name was an empty string.
ENOEQFLOG	3466	Object is already linked to a dead directory.	The link as a dead option was specified, but the object is already marked as dead. Only one dead link is allowed for an object.
EEMPTYDIR	3467	Directory is empty.	A directory with entries of only dot and dot-dot was supplied when a nonempty directory was expected.
EMLINK	3468	Maximum link count for a file was exceeded.	An attempt was made to have the link count of a single file exceed LINK_MAX. The value of LINK_MAX can be determined using the pathconf() or the fpathconf() function.
ESPIPE	3469	Seek request is not supported for object.	A seek request was specified for an object that does not support seeking.

Name	Value	Text	Details
ENOSYS	3470	Function not implemented.	An attempt was made to use a function that is not available in this implementation for any object or any arguments.
EISDIR	3471	Specified target is a directory.	The path specified named a directory where a file or object name was expected.
EROFS	3472	Read-only file system.	You have attempted an update operation in a file system that only supports read operations.
EC2	3473	C2 pointer validation error.	
EUNKNOWN	3474	Unknown system state.	The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.
EITERBAD	3475	Iterator is not valid.	
EITERSTE	3476	Iterator is in wrong state for operation.	
EHRICLSBAD	3477	HRI class is not valid.	
EHRICLBAD	3478	HRI subclass is not valid.	
EHRITYPBAD	3479	HRI type is not valid.	
ENOTAPPL	3480	Data requested is not applicable.	
EHRIREQTYP	3481	HRI request type is not valid.	
EHRINAMEBAD	3482	HRI resource name is not valid.	
EDAMAGE	3484	A damaged object was encountered.	
ELOOP	3485	A loop exists in the symbolic links.	This error is issued if the number of symbolic links encountered is more than POSIX_SYMLOOP (defined in the limits.h header file). Symbolic links are encountered during resolution of the directory or path name.
ENAMETOOLONG	3486	A path name is too long.	A path name is longer than PATH_MAX characters or some component of the name is longer than NAME_MAX characters while _POSIX_NO_TRUNC is in effect. For symbolic links, the length of the name string substituted for a symbolic link exceeds PATH_MAX. The PATH_MAX and NAME_MAX values can be determined using the **pathconf()** function.
ENOLCK	3487	No locks are available.	A system-imposed limit on the number of simultaneous file and record locks was reached, and no more were available at that time.
ENOTEMPTY	3488	Directory is not empty.	You tried to remove a directory that is not empty. A directory cannot contain objects when it is being removed.
ENOSYSRSC	3489	System resources are not available.	

Name	Value	Text	Details
ECONVERT	3490	Conversion error.	One or more characters could not be converted from the source CCSID to the target CCSID.
E2BIG	3491	Argument list is too long.	
EILSEQ	3492	Conversion stopped due to input character that does not belong to the input codeset.	
ETYPE	3493	Object type mismatch.	The type of the object referenced by a descriptor does not match the type specified on the interface.
EBADDIR	3494	Attempted to reference a directory that was not found or was destroyed.	
EBADOBJ	3495	Attempted to reference an object that was not found, was destroyed, or was damaged.	
EIDXINVAL	3496	Data space index used as a directory is not valid.	
ESOFTDAMAGE	3497	Object has soft damage.	
ENOTENROLL	3498	User is not enrolled in system distribution directory.	You attempted to use a function that requires you to be enrolled in the system distribution directory and you are not.
EOFFLINE	3499	Object is suspended.	You have attempted to use an object that has had its data saved and the storage associated with it freed. An attempt to retrieve the object's data failed. The object's data cannot be used until it is successfully restored. The object's data was saved and freed either by saving the object with the STG(*FREE) parameter, or by calling an API.
EROOBJ	3500	Object is read-only.	You have attempted to update an object that can be read only.
EEAHDDSI	3501	Hard damage on extended attribute data space index.	
EEASDDSI	3502	Soft damage on extended attribute data space index.	
EEAHDDS	3503	Hard damage on extended attribute data space.	
EEASDDS	3504	Soft damage on extended attribute data space.	
EEADUPRC	3505	Duplicate extended attribute record.	
ELOCKED	3506	Area being read from or written to is locked.	The read or write of an area conflicts with a lock held by another process.
EFBIG	3507	Object too large.	The size of the object would exceed the system allowed maximum size.
EIDRM	3509	The semaphore, shared memory, or message queue identifier is removed from the system.	

Name	Value	Text	Details
ENOMSG	3510	The queue does not contain a message of the desired type and (msgflg logically ANDed with IPC_NOWAIT).	
EFILECVT	3511	File ID conversion of a directory failed.	To recover from this error, run the Reclaim Storage (RCLSTG) command as soon as possible.
EBADFID	3512	A file ID could not be assigned when linking an object to a directory.	The file ID table is missing or damaged. To recover from this error, run the Reclaim Storage (RCLSTG) command as soon as possible.
ESTALE	3513	File or object handle rejected by server.	
ESRCH	3515	No such process.	
ENOTSIGINIT	3516	Process is not enabled for signals.	An attempt was made to call a signal function under one of the following conditions: • The signal function is being called for a process that is not enabled for asynchronous signals. • The signal function is being called when the system signal controls have not been initialized.
ECHILD	3517	No child process.	
EBADH	3520	Handle is not valid.	
ETOOMANYREFS	3523	The operation would have exceeded the maximum number of references allowed for a descriptor.	
ENOTSAFE	3524	Function is not allowed.	Function is not allowed in a job that is running with multiple threads.
EOVERFLOW	3525	Object is too large to process.	The object's data size exceeds the limit allowed by this function.
EJRNDAMAGE	3526	Journal is damaged.	A journal or all of the journal's attached journal receivers are damaged, or the journal sequence number has exceeded the maximum value allowed. This error occurs during operations that were attempting to send an entry to the journal.
EJRNINACTIVE	3527	Journal is inactive.	The journaling state for the journal is *INACTIVE. This error occurs during operations that were attempting to send an entry to the journal.
EJRNRCVSPC	3528	Journal space or system storage error.	The attached journal receiver does not have space for the entry because the storage limit has been exceeded for the system, the object, the user profile, or the group profile. This error occurs during operations that were attempting to send an entry to the journal.

Name	Value	Text	Details
EJRNRMT	3529	Journal is remote.	The journal is a remote journal. Journal entries cannot be sent to a remote journal. This error occurs during operations that were attempting to send an entry to the journal.
ENEWJRNRCV	3530	New journal receiver is needed.	A new journal receiver must be attached to the journal before entries can be journaled. This error occurs during operations that were attempting to send an entry to the journal.
ENEWJRN	3531	New journal is needed.	The journal was not completely created, or an attempt to delete it did not complete successfully. This error occurs during operations that were attempting to start or end journaling, or were attempting to send an entry to the journal.
EJOURNALED	3532	Object already journaled.	A start journaling operation was attempted on an object that is already being journaled.
EJRNENTTOOLONG	3533	Entry is too large to send.	The journal entry generated by this operation is too large to send to the journal.
EDATALINK	3534	Object is a datalink object.	
ENOTAVAIL	3535	Independent Auxiliary Storage Pool (ASP) is not available.	The independent ASP is in Vary Configuration (VRYCFG) or Reclaim Storage (RCLSTG) processing. To recover from this error, wait until processing has completed for the independent ASP.
ENOTTY	3536	I/O control operation is not appropriate.	
EFBIG2	3540	Attempt to write or truncate file past its sort file size limit.	
ETXTBSY	3543	Text file busy.	An attempt was made to execute an i5/OS® PASE program that is currently open for writing, or an attempt has been made to open for writing an i5/OS PASE program that is being executed.
EASPGRPNOTSET	3544	ASP group not set for thread.	
ERESTART	3545	A system call was interrupted and may be restarted.	
ESCANFAILURE	3546	Object had scan failure.	An object has been marked as a scan failure due to processing by an exit program associated with the scan-related integrated file system exit points.

Top | UNIX-Type APIs | APIs by category

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY  10504-1785
U.S.A.
```

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

```
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan
```

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

This API descriptions publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Advanced 36
Advanced Function Presentation
Advanced Peer-to-Peer Networking
AFP
AIX
AnyNet
AS/400
BCOCA
C/400
COBOL/400
Common User Access
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI
DRDA
Enterprise Storage Server
eServer
FlashCopy
GDDM
i5/OS
IBM
IBM (logo)
InfoColor
Infoprint
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
Lotus
Lotus Notes
MO:DCA
MVS
Net.Data
NetServer
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
POWER5+
PowerPC
Print Services Facility
PrintManager
PROFS
RISC System/6000
RPG/400
RS/6000

SAA
SecureWay
SOM
System i
System i5
System Object Model
System/36
System/38
System/390
TotalStorage
VisualAge
WebSphere
xSeries
z/OS

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER

EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Printed in USA