Note
Before using this information and the product it supports, be sure to read the information in "Notices," on page 37.
## Contents

### Environment Variable APIs

<table>
<thead>
<tr>
<th>APIs</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>getenv()—Get Value of Environment Variable</td>
<td>2</td>
</tr>
<tr>
<td>Parameters</td>
<td>2</td>
</tr>
<tr>
<td>Return Value</td>
<td>2</td>
</tr>
<tr>
<td>Error Conditions</td>
<td>2</td>
</tr>
<tr>
<td>Usage Notes</td>
<td>3</td>
</tr>
<tr>
<td>Related Information</td>
<td>3</td>
</tr>
<tr>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>putenv()—Change or Add Environment Variable</td>
<td>4</td>
</tr>
<tr>
<td>Parameters</td>
<td>4</td>
</tr>
<tr>
<td>Return Value</td>
<td>4</td>
</tr>
<tr>
<td>Error Conditions</td>
<td>5</td>
</tr>
<tr>
<td>Usage Notes</td>
<td>5</td>
</tr>
<tr>
<td>Related Information</td>
<td>5</td>
</tr>
<tr>
<td>Example</td>
<td>6</td>
</tr>
<tr>
<td>Qp0zDelEnv()—Delete an Environment Variable</td>
<td>7</td>
</tr>
<tr>
<td>Parameters</td>
<td>7</td>
</tr>
<tr>
<td>Authorities</td>
<td>7</td>
</tr>
<tr>
<td>Return Value</td>
<td>7</td>
</tr>
<tr>
<td>Error Conditions</td>
<td>7</td>
</tr>
<tr>
<td>Usage Notes</td>
<td>7</td>
</tr>
<tr>
<td>Related Information</td>
<td>7</td>
</tr>
<tr>
<td>Example</td>
<td>8</td>
</tr>
<tr>
<td>Qp0zDelSysEnv()—Delete a System-Level Environment Variable</td>
<td>9</td>
</tr>
<tr>
<td>Parameters</td>
<td>9</td>
</tr>
<tr>
<td>Authorities</td>
<td>9</td>
</tr>
<tr>
<td>Return Value</td>
<td>9</td>
</tr>
<tr>
<td>Error Conditions</td>
<td>10</td>
</tr>
<tr>
<td>Related Information</td>
<td>10</td>
</tr>
<tr>
<td>Example</td>
<td>11</td>
</tr>
<tr>
<td>Qp0zGetAllSysEnv()—Get All System-Level Environment Variables</td>
<td>11</td>
</tr>
<tr>
<td>Authorities</td>
<td>11</td>
</tr>
<tr>
<td>Parameters</td>
<td>11</td>
</tr>
<tr>
<td>Return Value</td>
<td>12</td>
</tr>
<tr>
<td>Error Conditions</td>
<td>12</td>
</tr>
<tr>
<td>Usage Notes</td>
<td>13</td>
</tr>
<tr>
<td>Related Information</td>
<td>13</td>
</tr>
<tr>
<td>Example</td>
<td>13</td>
</tr>
<tr>
<td>Qp0zGetEnv()—Get Value of Environment Variable (Extended)</td>
<td>13</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td>14</td>
</tr>
<tr>
<td>Parameters</td>
<td>14</td>
</tr>
<tr>
<td>Return Value</td>
<td>14</td>
</tr>
<tr>
<td>Error Conditions</td>
<td>14</td>
</tr>
<tr>
<td>Usage Notes</td>
<td>14</td>
</tr>
<tr>
<td>Related Information</td>
<td>15</td>
</tr>
<tr>
<td>Example</td>
<td>15</td>
</tr>
<tr>
<td>Qp0zGetSysEnv()—Get Value of System-Level Environment Variable</td>
<td>15</td>
</tr>
<tr>
<td>Authorities</td>
<td>15</td>
</tr>
<tr>
<td>Parameters</td>
<td>15</td>
</tr>
<tr>
<td>Return Value</td>
<td>16</td>
</tr>
<tr>
<td>Error Conditions</td>
<td>16</td>
</tr>
<tr>
<td>Usage Notes</td>
<td>17</td>
</tr>
<tr>
<td>Related Information</td>
<td>17</td>
</tr>
<tr>
<td>Example</td>
<td>17</td>
</tr>
<tr>
<td>Qp0zInitEnv()—Initialize Environment for Variables</td>
<td>17</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td>18</td>
</tr>
<tr>
<td>Parameters</td>
<td>18</td>
</tr>
<tr>
<td>Return Value</td>
<td>18</td>
</tr>
<tr>
<td>Error Conditions</td>
<td>18</td>
</tr>
<tr>
<td>Related Information</td>
<td>18</td>
</tr>
<tr>
<td>Example</td>
<td>18</td>
</tr>
<tr>
<td>Qp0zPutEnv()—Change or Add Environment Variable (Extended)</td>
<td>18</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td>19</td>
</tr>
<tr>
<td>Parameters</td>
<td>19</td>
</tr>
<tr>
<td>Return Value</td>
<td>19</td>
</tr>
<tr>
<td>Error Conditions</td>
<td>19</td>
</tr>
<tr>
<td>Usage Notes</td>
<td>20</td>
</tr>
<tr>
<td>Related Information</td>
<td>20</td>
</tr>
<tr>
<td>Example</td>
<td>20</td>
</tr>
<tr>
<td>Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable</td>
<td>21</td>
</tr>
<tr>
<td>Parameters</td>
<td>21</td>
</tr>
<tr>
<td>Authorities</td>
<td>21</td>
</tr>
<tr>
<td>Return Value</td>
<td>21</td>
</tr>
<tr>
<td>Error Conditions</td>
<td>21</td>
</tr>
<tr>
<td>Usage Notes</td>
<td>22</td>
</tr>
<tr>
<td>Related Information</td>
<td>22</td>
</tr>
<tr>
<td>Example</td>
<td>23</td>
</tr>
</tbody>
</table>

### Appendix. Notices

| Programming Interface Information | 38 |
| Trademarks | 39 |
| Terms and Conditions | 40 |

© Copyright IBM Corp. 1998, 2006
Environment Variable APIs

Environment variables are character strings of the form "name=value". There are two types of environment variables:

- **Job-level environment variables.** The job-level environment variables are stored in an environment space outside of the program associated with the job. They can be manipulated by using the `getenv()`, `putenv()`, `Qp0zDltEnv()`, `Qp0zGetEnv()`, `Qp0zInitEnv()`, and `Qp0zPutEnv()` APIs, as well as the CL commands ADDENVVAR, CHGENVVAR, RMVENVVAR, and WRKENVVAR. These variables exist for the duration of the job or until they are deleted. There is a limit of 4095 job-level environment variables.

- **System-level environment variables.** The system-level environment variables are stored in a global environment space that is persistent across IPLs and is not associated to a particular job. They can be manipulated by using the `Qp0zDltSysEnv()`, `Qp0zGetAllSysEnv()`, `Qp0zGetSysEnv()`, and `Qp0zPutSysEnv()` APIs, as well as the CL commands ADDENVVAR, CHGENVVAR, RMVENVVAR, and WRKENVVAR. These variables exist until they are deleted. There is a limit of 4095 system-level environment variables.

When a job calls one of the job-level environment variable APIs or CL commands for the first time, it inherits the system-level environment variables onto its job-level environment space. Any changes to job-level and system-level environment variables are then independent of one another.

The temporary space where the job-level environment variables are stored allows read and write access. Therefore, it is possible for the space to be corrupted. This could occur if a programmer accesses the space directly using the `environ` array rather than using the environment variable APIs. If the space is corrupted, subsequent calls using the APIs will have unpredictable results.

The environment variable APIs are:

- **`getenv()`—Get Value of Environment Variable** on page 2 (Get value of environment variable) searches the job-level environment list for a string of the form name=value, where name is the environment variable and value is the value of the variable.

- **`putenv()`—Change or Add Environment Variable** on page 4 (Change or add environment variable) sets the value of a job-level environment variable by changing an existing variable or creating a new one.

- **`Qp0zDltEnv()`—Delete an Environment Variable** on page 7 (Delete an environment variable) deletes a single job-level environment variable or deletes all environment variables from the current job.

- **`Qp0zDltSysEnv()`—Delete a System-Level Environment Variable** on page 9 (Delete a system-level environment variable) deletes a single system-level environment variable or deletes all system-level environment variables.

- **`Qp0zGetAllSysEnv()`—Get All System-Level Environment Variables** on page 11 (Get all system-level environment variables) fills in the list_buf with a list of all the system-level environment variables.

- **`Qp0zGetEnv()`—Get Value of Environment Variable (Extended)** on page 13 (Get value of environment variable (extended)) is an i5/OS extension to the standard `getenv()` function.

- **`Qp0zGetSysEnv()`—Get Value of System-Level Environment Variable** on page 15 (Get value of system-level environment variable) gets the value of a system-level environment variable name by searching the system-level environment variable list for a string of the form name=value.

- **`Qp0zInitEnv()`—Initialize Environment for Variables** on page 17 (Initialize environment for variables) sets the external variable `environ` to a pointer to the current environment list.

- **`Qp0zPutEnv()`—Change or Add Environment Variable (Extended)** on page 18 (Change or add environment variable (extended)) is an i5/OS extension to the standard `putenv()` function.
“Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable” on page 21 (Change or add a system-level environment variable) sets the value of a system-level environment variable by altering an existing variable or creating a new variable.

Note: These functions use header (include) files from the library QSYSINC, which is optionally installable. Make sure QSYSINC is installed on your system before using any of the functions. See “Header Files for UNIX-Type Functions” on page 24 for the file and member name of each header file.

APIs
These are the APIs for this category.

getenv()—Get Value of Environment Variable

Syntax
#include <stdlib.h>

char *getenv(const char *name);

Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threadsafe: Yes. See Usage Notes for more information.

The getenv() function searches the job-level environment list for a string of the form name=value, where name is the environment variable and value is the value of the variable.

The name parameter does not include the equal (=) symbol or the value of the environment variable name=value pair.

Authorities and Locks
None.

Parameters
name  (Input) The name of an environment variable.

Return Value
value  getenv() successfully found the environment string. The value returned is a pointer to the string containing the value for the specified name in the current environment.
NULL  getenv() could not find the environment string. The errno variable is set to indicate the error.

Error Conditions
If getenv() is not successful, errno indicates one of the following errors.

EDAMAGE
A damaged object was encountered.
A referenced object is damaged. The object cannot be used.

EFAULT

The address used for an argument is not correct.
In attempting to use an argument in a call, the system detected an address that is not valid.
While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[ENOENT]
No such path or directory.
The directory or a component of the path name specified does not exist.
A named file or directory does not exist or is an empty string.
No entry found for name specified.

[EUNKNOWN]
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.

Usage Notes
1. Although getenv() is threadsafe, if a thread calls an environment variable function while another thread is accessing an environment variable from the environ array the thread may see undefined results. The environ array can be accessed directly or by using a pointer returned from the getenv() or Qp0zGetEnv() functions. The environment contents are only protected during calls to the environment variable functions.
2. All environment variables are stored with an associated CCSID (coded character set identifier). Unless a different CCSID is specified, such as by using Qp0zPutEnv(), the default CCSID for the job is used as the CCSID associated with each environment variable string.
3. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information
- “putenv()—Change or Add Environment Variable” on page 4—Change or Add Environment Variable
- “Qp0zDltEnv()—Delete an Environment Variable” on page 7—Delete an Environment Variable
- “Qp0zDltSysEnv()—Delete a System-Level Environment Variable” on page 9—Delete a System-Level Environment Variable
- “Qp0zGetAllSysEnv()—Get All System-Level Environment Variables” on page 11—Get All System-Level Environment Variables
- “Qp0zGetEnv()—Get Value of Environment Variable (Extended)” on page 13—Get Value of Environment Variable (Extended)
- “Qp0zGetSysEnv()—Get Value of System-Level Environment Variable” on page 15—Get Value of System-Level Environment Variable
- “Qp0zInitEnv()—Initialize Environment for Variables” on page 17—Initialize Environment for Variables
- “Qp0zPutEnv()—Change or Add Environment Variable (Extended)” on page 18—Change or Add Environment Variable (Extended)
- “Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable” on page 21—Change or Add a System-Level Environment

Example
See Code disclaimer information for information pertaining to code examples.
See the example of using `getenv()` in "putenv()—Change or Add Environment Variable"—Change or Add Environment Variable.

For other examples, see the following:
- Using Environment Variables
- Using the Spawn Process and Wait for Child Process APIs
- Using the Spawn Process (using NLS-enabled path name)

API introduced: V3R6

putenv()—Change or Add Environment Variable

Syntax

```c
#include <stdlib.h>

int putenv(const char *string);
```

Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threading: Yes. See Usage Notes for more information.

The `putenv()` function sets the value of a job-level environment variable by changing an existing variable or creating a new one. The `string` parameter points to a string of the form `name=value`, where `name` is the environment variable and `value` is the new value for it.

The name cannot contain a blank. For example,

```c
PATH NAME=/my_lib/joe_user
```

is not valid because of the blank between `PATH` and `NAME`. The name can contain an equal (=) symbol, but the system interprets all characters following the first equal symbol as being the value of the environment variable. For example,

```c
PATH=NAME=/my_lib/joe_user
```

will result in a value of 'NAME=/my_lib/joe_user' for the variable `PATH`.

Authorities and Locks

None.

Parameters

`string`  (Input) A pointer to the `name=value` string.

Return Value

- 0  `putenv()` was successful.
- -1  `putenv()` was not successful. The `errno` variable is set to indicate the error.
Error Conditions
If `putenv()` is not successful, `errno` indicates one of the following errors.

**[EDAMAGE]**
- A damaged object was encountered.
- A referenced object is damaged. The object cannot be used.

**[EFAULT]**
The address used for an argument is not correct. In attempting to use an argument in a call, the system detected an address that is not valid.
While attempting to access a parameter passed to this function, the system detected an address that is not valid.

**[EINVAL]**
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.
An argument value is not valid, out of range, or NULL. For example, the string may not be in the correct format.

**[ENOMEM]**
Storage allocation request failed.
A function needed to allocate storage, but no storage is available.
There is not enough memory to perform the requested function. (There is a limit of 4095 environment variables per job.)

**[ENOMSG]**
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.

Usage Notes
1. Although `putenv()` is threadsafe, if a thread calls an environment variable function while another thread is accessing an environment variable from the `environ` array the thread may see undefined results. The `environ` array can be accessed directly or by using a pointer returned from the `getenv()` or `Qp0zGetEnv()` functions. The environment contents are only protected during calls to the environment variable functions.

2. All environment variables are stored with an associated CCSID (coded character set identifier). Because `putenv()` does not specify a CCSID, the default CCSID for the job is used as the CCSID associated with strings that are stored using `putenv()`.

3. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information
- “`getenv()`—Get Value of Environment Variable” on page 2
- “`Qp0zDltEnv()`—Delete an Environment Variable” on page 7
- “`Qp0zDltSysEnv()`—Delete a System-Level Environment Variable” on page 9
- “`Qp0zGetAllSysEnv()`—Get All System-Level Environment Variables” on page 11
- “`Qp0zSetEnv()`—Set Environment Variables” on page 4

Environment Variable APIs
The following example uses putenv() and getenv().

```c
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>

int main(int argc, char **argv)
{
    char *var1 = "PATH=/:/home/userid";
    char *name1 = "PATH";
    char *val1 = NULL;
    int rc;

    rc = putenv(var1);
    if (rc < 0) {
        printf("Error inserting <\%s> in environ, errno = \%d\n", var1, errno);
        return 1;
    }

    printf("<\%s> inserted in environ\n", var1);
    val1 = getenv(name1);
    if (val1 == NULL) {
        printf("Error retrieving <\%s> from environ, errno = \%d\n", name1, errno);
        return 1;
    }

    printf("<\%s> retrieved from environ, value is <\%s>\n", name1, val1);
    return 0;
}
```

Output:

<PATH=/:/home/userid> inserted in environ
<PATH> retrieved from environ, value is <\%s>

For other examples, see the following:

- **Using Environment Variables**
- **Using the Spawn Process and Wait for Child Process APIs**
- **Using the Spawn Process (using NLS-enabled path name)**

API introduced: V3R6
Qp0zDltEnv()—Delete an Environment Variable

Syntax
#include <qp0z1170.h>

int Qp0zDltEnv(const char *name);

Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threadsafe: Yes. See Usage Notes for more information.

The Qp0zDltEnv() function deletes a single job-level environment variable or deletes all environment variables from the current job. If the name parameter is NULL, all environment variables in the job are deleted.

The name parameter does not include the equal (=) symbol or the value of the environment variable name=value pair.

Parameters
name  (Input) A pointer to the name part of the environment variable name=value string.

Authorities
None.

Return Value
0      Qp0zDltEnv() was successful.
-1     Qp0zDltEnv() was not successful. The errno variable is set to indicate the error.

Error Conditions
If Qp0zDltEnv() is not successful, errno indicates one of the following errors.

[ENOENT]

No such path or directory.

The directory or a component of the path name specified does not exist.

A named file or directory does not exist or is an empty string.

The parameter name is not NULL and does not point to an environment variable name that currently exists in the environment list.

Usage Notes
1. Although Qp0zDltEnv() is threadsafe, if a thread calls an environment variable function while another thread is accessing an environment variable from the environ array the thread may see undefined results. The environ array can be accessed directly or by using a pointer returned from the getenv() or Qp0zGetEnv() functions. The environment contents are only protected during calls to the environment variable functions.

Related Information
- "getenv()—Get Value of Environment Variable” on page 2
- "putenv()—Change or Add Environment Variable” on page 4
Example
See Code disclaimer information for information pertaining to code examples.

The following example uses Qp0zDltEnv(), putenv() and the environ array.

```c
#include <stdio.h>
#include <errno.h>
#include <qp0z1170.h>
#include <stdlib.h>
extern char **environ;

#define ASSERT(x, y)  
   { if ((x)) { 
      printf("Assertion Failed: " #x 
            , Description: " y 
            , errno=%d", errno); 
      exit(EXIT_FAILURE); 
   } 
}

int main(int argc, char **argv)
{
   int rc=0;
   int e=0;
   printf("Enter Testcase - %\n\n", argv[0]);
   rc = putenv("PATH=/usr/bin:/home/me:%LIBL%"):
   ASSERT((rc == 0), "putenv(PATH)");
   rc = putenv("TEST0=42");
   ASSERT((rc == 0), "putenv(TEST0)");
   rc = putenv("TEST1=42");
   ASSERT((rc == 0), "putenv(TEST1)");
   printf("Before delete, these environment variables are set: \n\n");
   while (environ[e] != NULL) {
      printf(" %s\n", environ[e]);
      ++e;
   }
   printf("Delete the environment variables\n");
   rc = Qp0zDltEnv("TEST0");
   ASSERT((rc==0), "Qp0zDltEnv(TEST0)");
   rc = Qp0zDltEnv("TEST1");
   ASSERT((rc==0), "Qp0zDltEnv(TEST1)");
   printf("After delete, these environment variables are set: \n\n");
```
```c
int Qp0zDltSysEnv(const char *name, void *reserved);
```

**Syntax**

```c
#include <qp0z1170.h>
```

**Service Program Name:** QP0ZSYSE

**Default Public Authority:** *USE

**Threadsafe:** Yes

The `Qp0zDltSysEnv()` function deletes a single system-level environment variable or deletes all system-level environment variables. If the `name` parameter is NULL, all system-level environment variables are deleted.

The `name` parameter does not include the equal (=) symbol or the value part of the environment variable name=value pair.

**Parameters**

- `name` (Input) The name of the environment variable to delete.
- `reserved` (Input) Reserved for future use. Currently, the only value allowed is NULL.

**Authorities**

*JOBCTL* special authority is required to delete a system-level environment variable.

**Return Value**

- **0** `Qp0zDltSysEnv()` was successful.
- **errval** `Qp0zDltSysEnv()` was not successful. `errval` is set to indicate the error.
Error Conditions

If \texttt{Qp0zDltSysEnv()} is not successful, \texttt{errval} indicates one of the following errors.

\begin{verbatim}
[EFAULT]
  The address used for an argument is not correct.
  In attempting to use an argument in a call, the system detected an address that is not valid.
  While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[EINVAL]
  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.
  An argument value is not valid, out of range, or NULL.
  The value for the \texttt{reserved} parameter was not NULL.

[ENOENT]
  No such path or directory.
  The directory or a component of the path name specified does not exist.
  A named file or directory does not exist or is an empty string.
  The parameter \texttt{name} is not NULL and does not point to an environment variable name that currently exists in the environment list.

[EPERM]
  Operation not permitted.
  You must have appropriate privileges or be the owner of the object or other resource to do the requested operation.
  You must have *JOBCTL special authority to delete a system-level environment variable.

[EUNKNOWN]
  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.
\end{verbatim}

Related Information

- The \texttt{<qp0z1170.h>} file (see “Header Files for UNIX-Type Functions” on page 24)
- “getenv()—Get Value of Environment Variable” on page 2—Get Value of Environment Variable
- “putenv()—Change or Add Environment Variable” on page 4—Change or Add Environment Variable
- “Qp0zDltEnv()—Delete an Environment Variable” on page 7—Delete an Environment Variable
- “Qp0zGetAllSysEnv()—Get All System-Level Environment Variables” on page 11—Get All System-Level Environment Variables
- “Qp0zGetEnv()—Get Value of Environment Variable (Extended)” on page 13—Get Value of Environment Variable (Extended)
- “Qp0zGetSysEnv()—Get Value of System-Level Environment Variable” on page 15—Get Value of System-Level Environment Variable
- “Qp0zInitEnv()—Initialize Environment for Variables” on page 17—Initialize Environment for Variables
Example

See Code disclaimer information for information pertaining to code examples.

See the example of using Qp0zDltSysEnv() in “Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable” on page 21—Change or Add a System-Level Environment.

API introduced: V4R4

Qp0zGetAllSysEnv()—Get All System-Level Environment Variables

Syntax

```c
#include <qp0z1170.h>

int Qp0zGetAllSysEnv(char *list_buf, int *list_buf_size,
                      int *ccsid_buf, int *ccsid_buf_size,
                      void *reserved);
```

Service Program Name: QP0ZSYSE
Default Public Authority: *USE
Threadsafe: Yes

The Qp0zGetAllSysEnv() function fills in the list_buf with a list of all the system-level environment variables. The list consists of multiple null-terminated name=value strings followed by an ending null-terminator. The coded character set identifier (CCSID) associated with each name=value string is returned in the ccsid_buf buffer.

Authorities

None

Parameters

- **list_buf**
  (Input/Output) The address of the buffer to receive the null-terminated name=value list.

- **list_buf_size**
  (Input/Output) A pointer to an integer that contains the information about the size (in bytes) of the list_buf buffer. Before calling Qp0zGetAllSysEnv(), this parameter should be set to the size of list_buf. If the size of this parameter is large enough to receive the list, then this field will be set to the exact size of the list upon returning from Qp0zGetAllSysEnv(). If the size of this parameter is not large enough to receive the list, then this field will contain the exact size required and ENOSPC will be the return value. In this case, the list_buf is not modified.

- **ccsid_buf**
  (Input/Output) The address of the buffer to receive the CCSIDs of the environment variables. The order of the CCSIDs returned corresponds to the order of the variables returned in the list_buf.

- **ccsid_buf_size**
  (Input/Output) A pointer to an integer that contains the information about the size (in bytes) of the ccsid_buf buffer. Before calling Qp0zGetAllSysEnv(), this should be set to the size of
ccsid_buf. If this size is enough to receive the CCSID list, then this field will contain the exact size of the CCSIDs received upon returning from Qp0zGetAllSysEnv(). If this size is not enough to receive the CCSID list, then this field will contain the exact size required and ENOSPC will be the return value. In this case, the ccsid_buf is not modified.

reserved
(Input) Reserved for future use. Currently, the only allowed value is NULL.

Return Value

0 Qp0zGetAllSysEnv() was successful. The list_buf contains the null-terminated system-level environment variable strings, and the ccsid_buf contains the CCSID of each variable in the same order. The list_buf_size contains the exact size of the environment variable list, and the ccsid_buf_size contains the exact size of the CCSID list.

errval Qp0zGetAllSysEnv() was not successful. errval indicates the error.

Error Conditions
If Qp0zGetAllSysEnv() is not successful, errval indicates one of the following errors.

[EFAULT]
The address used for an argument is not correct.
In attempting to use an argument in a call, the system detected an address that is not valid.
While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[EINVAL]
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.
An argument value is not valid, out of range, or NULL.
The value for the reserved parameter was not NULL.

[ENOENT]
No such path or directory.
The directory or a component of the path name specified does not exist.
A named file or directory does not exist or is an empty string.
There were no system-level environment variables.

[ENOSPC]
No space 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.
Insufficient space remains to hold the intended file, directory, or link.
The size of the buffers to receive the list and the CCSIDs was not enough. The list_buf_size and ccsid_buf_size parameters indicate the exact size needed for the list_buf ccsid_buf respectively.

[EUNKNOWN]
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.

Usage Notes

1. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information

- The `<qp0z1170.h>` file (see “Header Files for UNIX-Type Functions” on page 24)
- “getenv()—Get Value of Environment Variable” on page 2—Get Value of Environment Variable
- “putenv()—Change or Add Environment Variable” on page 4—Change or Add Environment Variable
- “Qp0zDltEnv()—Delete an Environment Variable” on page 7—Delete an Environment Variable
- “Qp0zDltSysEnv()—Delete a System-Level Environment Variable” on page 9—Delete a System-Level Environment Variable
- “Qp0zGetEnv()—Get Value of Environment Variable (Extended)”—Get Value of Environment Variable (Extended)
- “Qp0zGetSysEnv()—Get Value of System-Level Environment Variable” on page 15—Get Value of System-Level Environment Variable
- “Qp0zInitEnv()—Initialize Environment for Variables” on page 17—Initialize Environment for Variables
- “Qp0zPutEnv()—Change or Add Environment Variable (Extended)” on page 18—Change or Add Environment Variable (Extended)
- “Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable” on page 21—Change or Add a System-Level Environment

Example

See Code disclaimer information for information pertaining to code examples.

1. See the example in “Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable” on page 21—Change or Add a System-Level Environment.
2. See the two-part example in Saving and Restoring System-Level Environment Variables in Examples: APIS.

API introduced: V4R4

Qp0zGetEnv()—Get Value of Environment Variable (Extended)

Syntax

```c
#include <qp0z1170.h>

char *Qp0zGetEnv(const char *name, int *ccsid);
```

Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threadsafe: Yes. See Usage Notes for more information.

The Qp0zGetEnv() function is an i5/OS extension to the standard getenv() function. Qp0zGetEnv() searches the job-level environment list for a string of the form name=value. The value and the CCSID (coded character set identifier) associated with the environment variable name are returned.
Authorities and Locks
None.

Parameters

\textit{name} \hspace{1em} \text{(Input)} \hspace{1em} The name of an environment variable.

\textit{ccsid} \hspace{1em} \text{(Output)} \hspace{1em} The CCSID for the named environment variable.

Return Value

\textit{value} \hspace{1em} \text{\texttt{Qp0zGetEnv}()} successfully found the environment string. The value returned is a pointer to the string containing the value for the specified \textit{name} in the current environment.

\text{\texttt{NULL}} \hspace{1em} \text{\texttt{Qp0zGetEnv}()} could not find the environment string. The \textit{errno} variable is set to indicate the error.

Error Conditions

If \texttt{Qp0zGetEnv}() is not successful, \textit{errno} indicates one of the following errors.

\[\text{EDAMAGE}\]
A damaged object was encountered.
A referenced object is damaged. The object cannot be used.

\[\text{EFAULT}\]
The address used for an argument is not correct.
In attempting to use an argument in a call, the system detected an address that is not valid.
While attempting to access a parameter passed to this function, the system detected an address that is not valid.

\[\text{ENOENT}\]
No such path or directory.
The directory or a component of the path name specified does not exist.
A named file or directory does not exist or is an empty string.
No entry found for name specified.

\[\text{EUNKNOWN}\]
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.

Usage Notes

1. Although \texttt{Qp0zGetEnv}() is threadsafe, if a thread calls an environment variable function while another thread is accessing an environment variable from the environ array the thread may see undefined results. The environ array can be accessed directly or by using a pointer returned from the \texttt{getenv}() or \texttt{Qp0zGetEnv}() functions. The environment contents are only protected during calls to the environment variable functions.

2. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.
The Qp0zGetSysEnv() function gets the value of a system-level environment variable name by searching the system-level environment variable list for a string of the form name=value. The value and the coded character set identifier (CCSID) associated with the environment variable name are returned.

**Authorities**
None

**Parameters**

- **name** (Input) The name of an environment variable.
- **value** (Input/Output) The address of the buffer to receive the value.
value_size
(Input/Output) A pointer to an integer that contains the information about the size of the value buffer. Before calling Qp0zGetSysEnv(), this parameter should contain the size of the value buffer. If the size of this parameter is large enough to receive the value, then this field will contain the exact size of value upon returning from Qp0zGetSysEnv(). If the size of this parameter is not large enough to receive the value, then this field will contain the exact size required and ENOSPC will be the return value. In this case, the value buffer is not modified.

ccsid
(Input/Output) The address of the variable to receive the CCSID associated with this variable.

reserved
(Input) Reserved for future use. Currently, the only allowed value is NULL.

Return Value
0 Qp0zGetSysEnv() successfully found the environment string. value and ccsid contain the value and CCSID for the variable name in the system-level environment variable list.
errval Qp0zGetEnv() was not successful. errval indicates the error.

Error Conditions
If Qp0zGetSysEnv() is not successful, errval indicates one of the following errors.

[EFAULT]
The address used for an argument is not correct.
In attempting to use an argument in a call, the system detected an address that is not valid.
While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[EINVAL]
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.
An argument value is not valid, out of range, or NULL.
The value for the reserved parameter was not NULL.

[ENOENT]
No such path or directory.
The directory or a component of the path name specified does not exist.
A named file or directory does not exist or is an empty string.
No entry found for name specified.

[ENOSPC]
No space 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.
Insufficient space remains to hold the intended file, directory, or link.
The size of the value buffer was not big enough to receive the value.

[EUNKNOWN]
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.

Usage Notes

1. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information

- The `<qp0z1170.h>` file (see “Header Files for UNIX-Type Functions” on page 24)
- “getenv()—Get Value of Environment Variable” on page 2—Get Value of Environment Variable
- “putenv()—Change or Add Environment Variable” on page 4—Change or Add Environment Variable
- “Qp0zDltEnv()—Delete an Environment Variable” on page 7—Delete an Environment Variable
- “Qp0zDltSysEnv()—Delete a System-Level Environment Variable” on page 9—Delete a System-Level Environment Variable
- “Qp0zGetAllSysEnv()—Get All System-Level Environment Variables” on page 11—Get All System-Level Environment Variables
- “Qp0zGetEnv()—Get Value of Environment Variable (Extended)” on page 13—Get Value of Environment Variable (Extended)
- “Qp0zInitEnv()—Initialize Environment for Variables”—Initialize Environment for Variables
- “Qp0zPutEnv()—Change or Add Environment Variable (Extended)” on page 18—Change or Add Environment Variable (Extended)
- “Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable” on page 21—Change or Add a System-Level Environment

Example

See Code disclaimer information for information pertaining to code examples.

See the example of using `Qp0zGetSysEnv()` in “Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable” on page 21—Change or Add a System-Level Environment.

API introduced: V4R4

Qp0zInitEnv()—Initialize Environment for Variables

Syntax

```c
#include <qp0z1170.h>

int Qp0zInitEnv(void);
```

Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threadsafe: Yes

The `Qp0zInitEnv()` function sets the external variable `environ` to a pointer to the current environment list. (On the iSeries server, `environ` is initialized to NULL when an activation group is started.)

Note: Although it is possible for a user’s program to directly read the `environ` array, use of the `getenv()` or `Qp0zGetEnv()` functions is recommended.
Authorities and Locks
None.

Parameters
None.

Return Value

0    Qp0zInitEnv() successfully initialized the environment.
-1    Qp0zInitEnv() was not successful. The errno variable is set to indicate the error.

Error Conditions
If Qp0zInitEnv() is not successful, errno indicates the following error.

[EUNKNOWN]
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.

Related Information
• The <qp0z1170.h> file (see "Header Files for UNIX-Type Functions" on page 24)
• “getenv()—Get Value of Environment Variable” on page 2—Get Value of Environment Variable
• “putenv()—Change or Add Environment Variable” on page 4—Change or Add Environment Variable
• “Qp0zDltEnv()—Delete an Environment Variable” on page 7—Delete an Environment Variable
• “Qp0zDltSysEnv()—Delete a System-Level Environment Variable” on page 9—Delete a System-Level Environment Variable
• “Qp0zGetAllSysEnv()—Get All System-Level Environment Variables” on page 11—Get All System-Level Environment Variables
• “Qp0zGetEnv()—Get Value of Environment Variable (Extended)” on page 13—Get Value of Environment Variable (Extended)
• “Qp0zGetSysEnv()—Get Value of System-Level Environment Variable” on page 15—Get Value of System-Level Environment Variable
• “Qp0zPutEnv()—Change or Add Environment Variable (Extended)”—Change or Add Environment Variable (Extended)
• “Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable” on page 21—Change or Add a System-Level Environment

API introduced: V3R6

---

Qp0zPutEnv()—Change or Add Environment Variable (Extended)

Syntax

```c
#include <qp0z1170.h>

int Qp0zPutEnv(const char *string, int ccsid);
```
Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threadsafe: Yes. See Usage Notes for more information.

The Qp0zPutEnv() function is an i5/OS extension to the standard putenv() function. Qp0zPutEnv() sets the value of an environment variable by altering an existing variable or creating a new variable. In addition, it specifies a CCSID (coded character set identifier) to be associated with the environment variable.

The string parameter points to a string of the form name=value, where name is the environment variable and value is the new value for it.

The name cannot contain a blank. For example,

```
PATH NAME=/my_lib/joe_user
```

is not valid because of the blank between PATH and NAME. The name can contain an equal (=) symbol, but the system interprets all characters following the first equal symbol as being the value of the environment variable. For example,

```
PATH=NAME=/my_lib/joe_user
```

will result in a value of 'NAME=/my_lib/joe_user' for the variable PATH.

**Authorities and Locks**
None.

**Parameters**

- **string** (Input) A pointer to the name=value string.
- **ccsid** (Input) A CCSID to be associated with this environment variable. If 0 is specified, the default CCSID for the job is used.

**Return Value**

- **0** Qp0zInitEnv() successfully initialized the environment.
- **-1** Qp0zInitEnv() was not successful. The errno variable is set to indicate the error.

**Error Conditions**
If Qp0zPutEnv() is not successful, errno indicates one of the following errors.

- **[EDAMAGE]**
  A damaged object was encountered.
  A referenced object is damaged. The object cannot be used.

- **[EFAULT]**
  The address used for an argument is not correct.
  In attempting to use an argument in a call, the system detected an address that is not valid.
  While attempting to access a parameter passed to this function, the system detected an address that is not valid.

- **[EINVAL]**
  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.

An argument value is not valid, out of range, or NULL. For example, the string may not be in the correct format.

\[\text{ENOMEM}\]

Storage allocation request failed.

A function needed to allocate storage, but no storage is available.

There is not enough memory to perform the requested function. (There is a limit of 4095 environment variables per job.)

\[\text{EUNKNOWN}\]

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.

**Usage Notes**

1. Although Qp0zPutEnv() is threadsafe, if a thread calls an environment variable function while another thread is accessing an environment variable from the environ array the thread may see undefined results. The environ array can be accessed directly or by using a pointer returned from the getenv() or Qp0zGetEnv() functions. The environment contents are only protected during calls to the environment variable functions.

2. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

**Related Information**

- The <qp0z1170.h> file (see "Header Files for UNIX-Type Functions" on page 24)
- "getenv()—Get Value of Environment Variable" on page 2—Get Value of Environment Variable
- "putenv()—Change or Add Environment Variable” on page 4—Change or Add Environment Variable
- "Qp0zDltEnv()—Delete an Environment Variable" on page 7—Delete an Environment Variable
- "Qp0zDltSysEnv()—Delete a System-Level Environment Variable” on page 9—Delete a System-Level Environment Variable
- "Qp0zGetAllSysEnv()—Get All System-Level Environment Variables” on page 11—Get All System-Level Environment Variables
- "Qp0zGetSysEnv()—Get Value of System-Level Environment Variable” on page 15—Get Value of System-Level Environment Variable
- "Qp0zInitEnv()—Initialize Environment for Variables” on page 17—Initialize Environment for Variables
- "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable” on page 21—Change or Add a System-Level Environment Variable

**Example**

See Code disclaimer information for information pertaining to code examples.

See the example of using putenv() in “Qp0zPutEnv()—Change or Add Environment Variable (Extended)” on page 18—Change or Add Environment Variable.

API introduced: V3R6
Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable

Syntax
#include <qp0z1170.h>

int Qp0zPutSysEnv(const char *string, int ccsid,
    void *reserved);

Service Program Name: QP0ZSYSE
Default Public Authority: *USE
Threadsafe: Yes

Qp0zPutSysEnv() function sets the value of a system-level environment variable by altering an existing variable or creating a new variable. In addition, it specifies a CCSID (coded character set identifier) to be associated with the environment variable.

The string parameter points to a string of the form name=value, where name is the environment variable and value is the new value for it.

The name cannot contain a blank. For example,

    PATH NAME=/my_lib/joe_user

is not valid because of the blank between PATH and NAME. The name can contain an equal (=) symbol, but the system interprets all characters following the first equal symbol as being the value of the environment variable. For example,

    PATH=NAME=/my_lib/joe_user

will result in a value of 'NAME=/my_lib/joe_user' for the variable PATH.

Parameters

string  (Input) A pointer to the name=value string.
ccsid  (Input) A CCSID to be associated with this environment variable. If 0 is specified, the default CCSID for the job is used.
reserved  (Input) Reserved for future use. Currently, the only allowed value is NULL.

Authorities

*JOBCTL special authority is required to add or change a system-level environment variable.

Return Value

0 Qp0zPutSysEnv() was successful.
errval Qp0zPutSysEnv() was not successful. errval is set to indicate the error.

Error Conditions

If Qp0zPutSysEnv() is not successful, errval indicates one of the following errors.

EFAULT

The address used for an argument is not correct.
In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[EINVAL]

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.
An argument value is not valid, out of range, or NULL.
For example, the string parameter was not in the correct format or the value for the reserved parameter was not NULL.

[ENOMEM]

Storage allocation request failed.
A function needed to allocate storage, but no storage is available.
There is not enough memory to perform the requested function. (There is a limit of 4095 system-level environment variables.)

[EOPNOTSUPP]

Operation not supported.
The operation, though supported in general, is not supported for the requested object or the requested arguments.
This error is returned if the environment variable that is being added is QIBM_CHILD_JOB_SNDINQMSG. See spawn() in or spawnp() in for details on QIBM_CHILD_JOB_SNDINQMSG.

[EPERM]

Operation not permitted.
You must have appropriate privileges or be the owner of the object or other resource to do the requested operation.
You must have *JOBCTL special authority to add or change system-level environment variables.

[EUNKNOWN]

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.

Usage Notes
1. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information
- The <qp0z1170.h> file (see "Header Files for UNIX-Type Functions" on page 24)
- "getenv()—Get Value of Environment Variable" on page 2—Get Value of Environment Variable
- "putenv()—Change or Add Environment Variable” on page 4—Change or Add Environment Variable
- “Qp0zDltEnv()—Delete an Environment Variable” on page 7—Delete an Environment Variable
- “Qp0zDltSysEnv()—Delete a System-Level Environment Variable” on page 9—Delete a System-Level Environment Variable
Example

See Code disclaimer information for information pertaining to code examples.

The following example uses Qp0zPutSysEnv(), Qp0zGetSysEnv(), and Qp0zDltSysEnv().

```c
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <qp0z1170.h>

int main(int argc, char **argv)
{
    char *var1 = "PATH=/:/home";
    char *name1 = "PATH";
    char *val1 = NULL;
    int rc, ccsid, size;

    /* Add the system-level variable PATH */
    /* using default ccsid */
    ccsid = 0;
    rc = Qp0zPutSysEnv(var1, ccsid, NULL);
    if(rc != 0)
    {
        printf("Error from Qp0zPutSysEnv while adding <%s>\n",var1);
        printf("errno = %d\n",rc);
        return rc;
    }

    printf("<%s> added to system-level env var list\n",var1);

    /* Get the value of the variable PATH */
    size = 100;
    val1 = (char *)malloc(size);

    rc = Qp0zGetSysEnv(name1, val1, &size, &ccsid, NULL);
    if(rc == ENOSPC)
    {
        /* The buffer size was not enough to get the value */
        /* Increase the buffer to size */
        val1 = (char *)realloc(val1, size);
        rc = Qp0zGetSysEnv(name1, val1, &size, &ccsid, NULL);
    }
    if(rc != 0)
    {
        printf("Error from Qp0zGetSysEnv while retrieving\n");
        printf("<%s>, errno = %d\n", name1, rc);
        return rc;
    }

    printf("<%s> retrieved, value is <%s>\n",name1,val1);

    /* Delete the PATH variable */
```
rc = Qp0zDltSysEnv(name1, NULL);
if(rc != 0) {
    printf("Error from Qp0zDltSysEnv while deleting\n");
    printf("%s, errno = %d\n", name1, rc);
    return rc;
}
printf("%s\n" deleted from system-level env var list\n", name1);
return 0;

Output:
<PATH=/:/home> added to system-level variable list
<PATH> retrieved, value is </:/home>
<PATH> deleted from system-level variable list

For other examples, see the two-part example in API Examples for saving and restoring system-level environment variables.

API introduced: V4R4
<table>
<thead>
<tr>
<th>Name of Header File</th>
<th>Name of File in QSYSINC</th>
<th>Name of Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>fcntl.h</td>
<td>H</td>
<td>FCNTL</td>
</tr>
<tr>
<td>grp.h</td>
<td>H</td>
<td>GRP</td>
</tr>
<tr>
<td>inttypes.h</td>
<td>H</td>
<td>INTTYPES</td>
</tr>
<tr>
<td>limits.h</td>
<td>H</td>
<td>LIMITS</td>
</tr>
<tr>
<td>mman.h</td>
<td>H</td>
<td>MMAN</td>
</tr>
<tr>
<td>netdbh.h</td>
<td>H</td>
<td>NETDB</td>
</tr>
<tr>
<td>netinet/icmp6.h</td>
<td>NETINET</td>
<td>ICMP6</td>
</tr>
<tr>
<td>net/if.h</td>
<td>NET</td>
<td>IF</td>
</tr>
<tr>
<td>netinet/in.h</td>
<td>NETINET</td>
<td>IN</td>
</tr>
<tr>
<td>netinet/ip_icmp.h</td>
<td>NETINET</td>
<td>IP_ICMP</td>
</tr>
<tr>
<td>netinet/ip.h</td>
<td>NETINET</td>
<td>IP</td>
</tr>
<tr>
<td>netinet/ip6.h</td>
<td>NETINET</td>
<td>IP6</td>
</tr>
<tr>
<td>netinet/tcp.h</td>
<td>NETINET</td>
<td>TCP</td>
</tr>
<tr>
<td>netinet/udp.h</td>
<td>NETINET</td>
<td>UDP</td>
</tr>
<tr>
<td>netns/idp.h</td>
<td>NETNS</td>
<td>IDP</td>
</tr>
<tr>
<td>netns/ixs.h</td>
<td>NETNS</td>
<td>IPX</td>
</tr>
<tr>
<td>netns/ns.h</td>
<td>NETNS</td>
<td>NS</td>
</tr>
<tr>
<td>netns/sp.h</td>
<td>NETNS</td>
<td>SP</td>
</tr>
<tr>
<td>net/route.h</td>
<td>NET</td>
<td>ROUTE</td>
</tr>
<tr>
<td>nettel/tel.h</td>
<td>NETTEL</td>
<td>TEL</td>
</tr>
<tr>
<td>os2.h</td>
<td>H</td>
<td>OS2</td>
</tr>
<tr>
<td>os2def.h</td>
<td>H</td>
<td>OS2DEF</td>
</tr>
<tr>
<td>pwd.h</td>
<td>H</td>
<td>PWD</td>
</tr>
<tr>
<td>Qlg.h</td>
<td>H</td>
<td>QLG</td>
</tr>
<tr>
<td>qp0chsg.h</td>
<td>H</td>
<td>QP0LCHSG</td>
</tr>
<tr>
<td>qp0flop.h</td>
<td>H</td>
<td>QP0LFLOP</td>
</tr>
<tr>
<td>qp0ljrn1.h</td>
<td>H</td>
<td>QP0LJRN1</td>
</tr>
<tr>
<td>qp0lror.h</td>
<td>H</td>
<td>QP0LROR</td>
</tr>
<tr>
<td>qp0lrrro.h</td>
<td>H</td>
<td>QP0LRRRO</td>
</tr>
<tr>
<td>qp0lrtsg.h</td>
<td>H</td>
<td>QP0LRTSG</td>
</tr>
<tr>
<td>qp0lscan.h</td>
<td>H</td>
<td>QP0LSCAN</td>
</tr>
<tr>
<td>Qp0lstdi.h</td>
<td>H</td>
<td>QP0LSTDI</td>
</tr>
<tr>
<td>qp0wpid.h</td>
<td>H</td>
<td>QP0WPID</td>
</tr>
<tr>
<td>qp0zdipc.h</td>
<td>H</td>
<td>QP0ZDIPC</td>
</tr>
<tr>
<td>qp0zipc.h</td>
<td>H</td>
<td>QP0ZIPC</td>
</tr>
<tr>
<td>qp0zolip.h</td>
<td>H</td>
<td>QP0ZOLIP</td>
</tr>
<tr>
<td>qp0zolsm.h</td>
<td>H</td>
<td>QP0ZOLSM</td>
</tr>
<tr>
<td>qp0zrpc.h</td>
<td>H</td>
<td>QP0ZRIPC</td>
</tr>
<tr>
<td>qp0ztrc.h</td>
<td>H</td>
<td>QP0ZTRC</td>
</tr>
<tr>
<td>qp0ztrml.h</td>
<td>H</td>
<td>QP0ZTRML</td>
</tr>
<tr>
<td>qp0z1170.h</td>
<td>H</td>
<td>QP0Z1170</td>
</tr>
<tr>
<td>Name of Header File</td>
<td>Name of File in QSYSINC</td>
<td>Name of Member</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>qsoasync.h</td>
<td>H</td>
<td>QSOASYNC</td>
</tr>
<tr>
<td>qtnxaapi.h</td>
<td>H</td>
<td>QTNXAAPI</td>
</tr>
<tr>
<td>qtnxadtp.h</td>
<td>H</td>
<td>QTNXADTP</td>
</tr>
<tr>
<td>qtomeapi.h</td>
<td>H</td>
<td>QTOMEAPI</td>
</tr>
<tr>
<td>qtoossapi.h</td>
<td>H</td>
<td>QTOSSAPI</td>
</tr>
<tr>
<td>resolv.h</td>
<td>H</td>
<td>RESOLVE</td>
</tr>
<tr>
<td>semaphore.h</td>
<td>H</td>
<td>SEMAPHORE</td>
</tr>
<tr>
<td>signal.h</td>
<td>H</td>
<td>SIGNAL</td>
</tr>
<tr>
<td>spawn.h</td>
<td>H</td>
<td>SPAWN</td>
</tr>
<tr>
<td>ssl.h</td>
<td>H</td>
<td>SSL</td>
</tr>
<tr>
<td>sys/errno.h</td>
<td>H</td>
<td>ERRNO</td>
</tr>
<tr>
<td>sys/ioctl.h</td>
<td>SYS</td>
<td>IOCTL</td>
</tr>
<tr>
<td>sys/ipc.h</td>
<td>SYS</td>
<td>IPC</td>
</tr>
<tr>
<td>sys/layout.h</td>
<td>H</td>
<td>LAYOUT</td>
</tr>
<tr>
<td>sys/limits.h</td>
<td>H</td>
<td>LIMITS</td>
</tr>
<tr>
<td>sys/msg.h</td>
<td>SYS</td>
<td>MSG</td>
</tr>
<tr>
<td>sys/param.h</td>
<td>SYS</td>
<td>PARAM</td>
</tr>
<tr>
<td>sys/resource.h</td>
<td>SYS</td>
<td>RESOURCE</td>
</tr>
<tr>
<td>sys/sem.h</td>
<td>SYS</td>
<td>SEM</td>
</tr>
<tr>
<td>sys/setParameter.h</td>
<td>SYS</td>
<td>SETJMP</td>
</tr>
<tr>
<td>sys/shm.h</td>
<td>SYS</td>
<td>SHM</td>
</tr>
<tr>
<td>sys/signal.h</td>
<td>SYS</td>
<td>SIGNAL</td>
</tr>
<tr>
<td>sys/socket.h</td>
<td>SYS</td>
<td>SOCKET</td>
</tr>
<tr>
<td>sys/stat.h</td>
<td>SYS</td>
<td>STAT</td>
</tr>
<tr>
<td>sys/statvfs.h</td>
<td>SYS</td>
<td>STATVFS</td>
</tr>
<tr>
<td>sys/time.h</td>
<td>SYS</td>
<td>TIME</td>
</tr>
<tr>
<td>sys/types.h</td>
<td>SYS</td>
<td>TYPES</td>
</tr>
<tr>
<td>sys/uio.h</td>
<td>SYS</td>
<td>UIO</td>
</tr>
<tr>
<td>sys/un.h</td>
<td>SYS</td>
<td>UN</td>
</tr>
<tr>
<td>sys/wait.h</td>
<td>SYS</td>
<td>WAIT</td>
</tr>
<tr>
<td>ulimit.h</td>
<td>H</td>
<td>ULIMIT</td>
</tr>
<tr>
<td>unistd.h</td>
<td>H</td>
<td>UNISTD</td>
</tr>
<tr>
<td>utime.h</td>
<td>H</td>
<td>UTIME</td>
</tr>
</tbody>
</table>

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:
  
  ```sql
  STRSEU SRCPF(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:
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 SRCPFILE(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`.

---

### Errno Values for UNIX-Type Functions

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Text</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDOM</td>
<td>3001</td>
<td>A domain error occurred in a math function.</td>
<td></td>
</tr>
<tr>
<td>ERANGE</td>
<td>3002</td>
<td>A range error occurred.</td>
<td></td>
</tr>
<tr>
<td>ETRUNC</td>
<td>3003</td>
<td>Data was truncated on an input, output, or update operation.</td>
<td></td>
</tr>
<tr>
<td>ENOTOPEN</td>
<td>3004</td>
<td>File is not open.</td>
<td>You attempted to do an operation that required the file to be open.</td>
</tr>
<tr>
<td>ENOTREAD</td>
<td>3005</td>
<td>File is not opened for read operations.</td>
<td>You tried to read a file that is not open for read operations.</td>
</tr>
<tr>
<td>EIO</td>
<td>3006</td>
<td>Input/output error.</td>
<td>A physical I/O error occurred or a referenced object was damaged.</td>
</tr>
<tr>
<td>ENODEV</td>
<td>3007</td>
<td>No such device.</td>
<td></td>
</tr>
<tr>
<td>ERECIO</td>
<td>3008</td>
<td>Cannot get single character for files opened for record I/O.</td>
<td>The file that was specified is open for record I/O and you attempted to read it as a stream file.</td>
</tr>
<tr>
<td>ENOTWRITE</td>
<td>3009</td>
<td>File is not opened for write operations.</td>
<td>You tried to update a file that has not been opened for write operations.</td>
</tr>
<tr>
<td>ESTDIN</td>
<td>3010</td>
<td>The stdin stream cannot be opened.</td>
<td></td>
</tr>
<tr>
<td>ESTDOUT</td>
<td>3011</td>
<td>The stdout stream cannot be opened.</td>
<td></td>
</tr>
<tr>
<td>ESTDERR</td>
<td>3012</td>
<td>The stderr stream cannot be opened.</td>
<td></td>
</tr>
<tr>
<td>EBADSEEK</td>
<td>3013</td>
<td>The positioning parameter in <code>seek</code> is not correct.</td>
<td></td>
</tr>
<tr>
<td>EBADNAME</td>
<td>3014</td>
<td>The object name specified is not correct.</td>
<td></td>
</tr>
<tr>
<td>EBADMODE</td>
<td>3015</td>
<td>The type variable specified on the open function is not correct.</td>
<td>The mode that you attempted to open the file in is not correct.</td>
</tr>
<tr>
<td>EBADPOS</td>
<td>3017</td>
<td>The position specifier is not correct.</td>
<td></td>
</tr>
<tr>
<td>ENOPOS</td>
<td>3018</td>
<td>There is no record at the specified position.</td>
<td>You attempted to position to a record that does not exist in the file.</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENUMMBRS</td>
<td>3019</td>
<td>Attempted to use ftell on multiple members.</td>
<td>Remove all but one member from the file.</td>
</tr>
<tr>
<td>ENUMRECS</td>
<td>3020</td>
<td>The current record position is too long for ftell.</td>
<td></td>
</tr>
<tr>
<td>EINVAL</td>
<td>3021</td>
<td>The value specified for the argument is not correct.</td>
<td>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.</td>
</tr>
<tr>
<td>EBADFUNC</td>
<td>3022</td>
<td>Function parameter in the signal function is not set.</td>
<td></td>
</tr>
<tr>
<td>ENOENT</td>
<td>3025</td>
<td>No such path or directory.</td>
<td>The directory or a component of the path name specified does not exist.</td>
</tr>
<tr>
<td>ENOREC</td>
<td>3026</td>
<td>Record is not found.</td>
<td></td>
</tr>
<tr>
<td>EPERM</td>
<td>3027</td>
<td>The operation is not permitted.</td>
<td>You must have appropriate privileges or be the owner of the object or other resource to do the requested operation.</td>
</tr>
<tr>
<td>EBADDATA</td>
<td>3028</td>
<td>Message data is not valid.</td>
<td>The message data that was specified for the error text is not correct.</td>
</tr>
<tr>
<td>EBUSY</td>
<td>3029</td>
<td>Resource busy.</td>
<td>An attempt was made to use a system resource that is not available at this time.</td>
</tr>
<tr>
<td>EBADEMT</td>
<td>3040</td>
<td>Option specified is not valid.</td>
<td></td>
</tr>
<tr>
<td>ENOTUPD</td>
<td>3041</td>
<td>File is not opened for update operations.</td>
<td></td>
</tr>
<tr>
<td>ENOTDLT</td>
<td>3042</td>
<td>File is not opened for delete operations.</td>
<td></td>
</tr>
<tr>
<td>EPAD</td>
<td>3043</td>
<td>The number of characters written is shorter than the expected record length.</td>
<td>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.</td>
</tr>
<tr>
<td>EBADKEYLN</td>
<td>3044</td>
<td>A length that was not valid was specified for the key.</td>
<td>You attempted a record I/O against a keyed file. The key length that was specified is not correct.</td>
</tr>
<tr>
<td>EPUTANDGET</td>
<td>3080</td>
<td>A read operation should not immediately follow a write operation.</td>
<td></td>
</tr>
<tr>
<td>EGETANDPUT</td>
<td>3081</td>
<td>A write operation should not immediately follow a read operation.</td>
<td></td>
</tr>
<tr>
<td>EIOERROR</td>
<td>3101</td>
<td>A nonrecoverable I/O error occurred.</td>
<td></td>
</tr>
<tr>
<td>EIORECERR</td>
<td>3102</td>
<td>A recoverable I/O error occurred.</td>
<td></td>
</tr>
<tr>
<td>EACCES</td>
<td>3401</td>
<td>Permission denied.</td>
<td>An attempt was made to access an object in a way forbidden by its object access permissions.</td>
</tr>
<tr>
<td>ENOTDIR</td>
<td>3403</td>
<td>Not a directory.</td>
<td>A component of the specified path name existed, but it was not a directory when a directory was expected.</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-----------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENOSPC</td>
<td>3404</td>
<td>No space is available.</td>
<td>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.</td>
</tr>
<tr>
<td>EXDEV</td>
<td>3405</td>
<td>Improper link.</td>
<td>A link to a file on another file system was attempted.</td>
</tr>
<tr>
<td>EAGAIN</td>
<td>3406</td>
<td>Operation would have caused the process to be suspended.</td>
<td></td>
</tr>
<tr>
<td>EW OULD BLOCK</td>
<td>3406</td>
<td>Operation would have caused the process to be suspended.</td>
<td></td>
</tr>
<tr>
<td>EINTR</td>
<td>3407</td>
<td>Interrupted function call.</td>
<td></td>
</tr>
<tr>
<td>EFAULT</td>
<td>3408</td>
<td>The address used for an argument was not correct.</td>
<td>In attempting to use an argument in a call, the system detected an address that is not valid.</td>
</tr>
<tr>
<td>ETIME</td>
<td>3409</td>
<td>Operation timed out.</td>
<td></td>
</tr>
<tr>
<td>ENXIO</td>
<td>3415</td>
<td>No such device or address.</td>
<td></td>
</tr>
<tr>
<td>EAPAR</td>
<td>3418</td>
<td>Possible APAR condition or hardware failure.</td>
<td></td>
</tr>
<tr>
<td>ERECURSE</td>
<td>3419</td>
<td>Recursive attempt rejected.</td>
<td></td>
</tr>
<tr>
<td>EADDRINUSE</td>
<td>3420</td>
<td>Address already in use.</td>
<td></td>
</tr>
<tr>
<td>EADDRNOTAVAIL</td>
<td>3421</td>
<td>Address is not available.</td>
<td></td>
</tr>
<tr>
<td>EAFNOSUPPORT</td>
<td>3422</td>
<td>The type of socket is not supported in this protocol family.</td>
<td></td>
</tr>
<tr>
<td>EALREADY</td>
<td>3423</td>
<td>Operation is already in progress.</td>
<td></td>
</tr>
<tr>
<td>ECONNABORTED</td>
<td>3424</td>
<td>Connection ended abnormally.</td>
<td></td>
</tr>
<tr>
<td>ECONNREFUSED</td>
<td>3425</td>
<td>A remote host refused an attempted connect operation.</td>
<td></td>
</tr>
<tr>
<td>ECONNRESET</td>
<td>3426</td>
<td>A connection with a remote socket was reset by that socket.</td>
<td></td>
</tr>
<tr>
<td>EDESTADDRREQ</td>
<td>3427</td>
<td>Operation requires destination address.</td>
<td></td>
</tr>
<tr>
<td>EHOSTDOW N</td>
<td>3428</td>
<td>A remote host is not available.</td>
<td></td>
</tr>
<tr>
<td>EHOSTUNREACH</td>
<td>3429</td>
<td>A route to the remote host is not available.</td>
<td></td>
</tr>
<tr>
<td>EINPROGRESS</td>
<td>3430</td>
<td>Operation in progress.</td>
<td></td>
</tr>
<tr>
<td>EISCONN</td>
<td>3431</td>
<td>A connection has already been established.</td>
<td></td>
</tr>
<tr>
<td>EMSGSIZE</td>
<td>3432</td>
<td>Message size is out of range.</td>
<td></td>
</tr>
<tr>
<td>ENETDOWN</td>
<td>3433</td>
<td>The network currently is not available.</td>
<td></td>
</tr>
<tr>
<td>ENETRESET</td>
<td>3434</td>
<td>A socket is connected to a host that is no longer available.</td>
<td></td>
</tr>
<tr>
<td>ENETUNREACH</td>
<td>3435</td>
<td>Cannot reach the destination network.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENOBUFFS</td>
<td>3436</td>
<td>There is not enough buffer space for the requested operation.</td>
<td></td>
</tr>
<tr>
<td>ENOPROTOOPT</td>
<td>3437</td>
<td>The protocol does not support the specified option.</td>
<td></td>
</tr>
<tr>
<td>ENOTCONN</td>
<td>3438</td>
<td>Requested operation requires a connection.</td>
<td></td>
</tr>
<tr>
<td>ENOTSOCK</td>
<td>3439</td>
<td>The specified descriptor does not reference a socket.</td>
<td></td>
</tr>
<tr>
<td>ENOTSUP</td>
<td>3440</td>
<td>Operation is not supported.</td>
<td>The operation, though supported in general, is not supported for the requested object or the requested arguments.</td>
</tr>
<tr>
<td>EOPNOTSUPP</td>
<td>3440</td>
<td>Operation is not supported.</td>
<td>The operation, though supported in general, is not supported for the requested object or the requested arguments.</td>
</tr>
<tr>
<td>EPFNOSUPPORT</td>
<td>3441</td>
<td>The socket protocol family is not supported.</td>
<td></td>
</tr>
<tr>
<td>EPROTONOSUPPORT</td>
<td>3442</td>
<td>No protocol of the specified type and domain exists.</td>
<td></td>
</tr>
<tr>
<td>EPROTOTYPE</td>
<td>3443</td>
<td>The socket type or protocols are not compatible.</td>
<td></td>
</tr>
<tr>
<td>ERCVDERR</td>
<td>3444</td>
<td>An error indication was sent by the peer program.</td>
<td></td>
</tr>
<tr>
<td>ESHUTDOWN</td>
<td>3445</td>
<td>Cannot send data after a shutdown.</td>
<td></td>
</tr>
<tr>
<td>ESOCKTNOSUPPORT</td>
<td>3446</td>
<td>The specified socket type is not supported.</td>
<td></td>
</tr>
<tr>
<td>ETIMEDOUT</td>
<td>3447</td>
<td>A remote host did not respond within the timeout period.</td>
<td></td>
</tr>
<tr>
<td>EUNATCH</td>
<td>3448</td>
<td>The protocol required to support the specified address family is not available at this time.</td>
<td></td>
</tr>
<tr>
<td>EBADF</td>
<td>3450</td>
<td>Descriptor is not valid.</td>
<td>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.</td>
</tr>
<tr>
<td>EMFILE</td>
<td>3452</td>
<td>Too many open files for this process.</td>
<td>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.</td>
</tr>
<tr>
<td>ENFILE</td>
<td>3453</td>
<td>Too many open files in the system.</td>
<td>A system limit has been reached for the number of files that are allowed to be concurrently open in the system.</td>
</tr>
<tr>
<td>EPIPE</td>
<td>3455</td>
<td>Broken pipe.</td>
<td></td>
</tr>
<tr>
<td>ECANCEL</td>
<td>3456</td>
<td>Operation cancelled.</td>
<td></td>
</tr>
<tr>
<td>EEXIST</td>
<td>3457</td>
<td>Object exists.</td>
<td>The object specified already exists and the specified operation requires that it not exist.</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>----------------</td>
<td>-------</td>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EDEADLK</td>
<td>3459</td>
<td>Resource deadlock avoided.</td>
<td>An attempt was made to lock a system resource that would have resulted in a deadlock situation. The lock was not obtained.</td>
</tr>
<tr>
<td>ENOMEM</td>
<td>3460</td>
<td>Storage allocation request failed.</td>
<td>A function needed to allocate storage, but no storage is available.</td>
</tr>
<tr>
<td>EOWNERTERM</td>
<td>3462</td>
<td>The synchronization object no longer exists because the owner is no longer running.</td>
<td>The process that had locked the mutex is no longer running, so the mutex was deleted.</td>
</tr>
<tr>
<td>EDESTROYED</td>
<td>3463</td>
<td>The synchronization object was destroyed, or the object no longer exists.</td>
<td></td>
</tr>
<tr>
<td>ETERM</td>
<td>3464</td>
<td>Operation was terminated.</td>
<td></td>
</tr>
<tr>
<td>ENOENT1</td>
<td>3465</td>
<td>No such file or directory.</td>
<td>A component of a specified path name did not exist, or the path name was an empty string.</td>
</tr>
<tr>
<td>ENOEQFLOG</td>
<td>3466</td>
<td>Object is already linked to a dead directory.</td>
<td>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.</td>
</tr>
<tr>
<td>EEMPTYDIR</td>
<td>3467</td>
<td>Directory is empty.</td>
<td>A directory with entries of only dot and dot-dot was supplied when a nonempty directory was expected.</td>
</tr>
<tr>
<td>EMLINK</td>
<td>3468</td>
<td>Maximum link count for a file was exceeded.</td>
<td>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.</td>
</tr>
<tr>
<td>ESPPIPE</td>
<td>3469</td>
<td>Seek request is not supported for object.</td>
<td>A seek request was specified for an object that does not support seeking.</td>
</tr>
<tr>
<td>ENOSYS</td>
<td>3470</td>
<td>Function not implemented.</td>
<td>An attempt was made to use a function that is not available in this implementation for any object or any arguments.</td>
</tr>
<tr>
<td>EISDIR</td>
<td>3471</td>
<td>Specified target is a directory.</td>
<td>The path specified named a directory where a file or object name was expected.</td>
</tr>
<tr>
<td>EROFS</td>
<td>3472</td>
<td>Read-only file system.</td>
<td>You have attempted an update operation in a file system that only supports read operations.</td>
</tr>
<tr>
<td>EUNKNOWN</td>
<td>3474</td>
<td>Unknown system state.</td>
<td>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.</td>
</tr>
<tr>
<td>EITERBAD</td>
<td>3475</td>
<td>Iterator is not valid.</td>
<td></td>
</tr>
<tr>
<td>EITERSTE</td>
<td>3476</td>
<td>Iterator is in wrong state for operation.</td>
<td></td>
</tr>
<tr>
<td>EHRICLSBAD</td>
<td>3477</td>
<td>HRI class is not valid.</td>
<td></td>
</tr>
<tr>
<td>EHRICLBAD</td>
<td>3478</td>
<td>HRI subclass is not valid.</td>
<td></td>
</tr>
<tr>
<td>EHRITYPBAD</td>
<td>3479</td>
<td>HRI type is not valid.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENOTAPPL</td>
<td>3480</td>
<td>Data requested is not applicable.</td>
<td></td>
</tr>
<tr>
<td>EHIRREQTYP</td>
<td>3481</td>
<td>HRI request type is not valid.</td>
<td></td>
</tr>
<tr>
<td>EHRINAMEBAD</td>
<td>3482</td>
<td>HRI resource name is not valid.</td>
<td></td>
</tr>
<tr>
<td>EDAMAGE</td>
<td>3484</td>
<td>A damaged object was encountered.</td>
<td>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.</td>
</tr>
<tr>
<td>ELOOP</td>
<td>3485</td>
<td>A loop exists in the symbolic links.</td>
<td></td>
</tr>
<tr>
<td>ENAMETOOLONG</td>
<td>3486</td>
<td>A path name is too long.</td>
<td>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.</td>
</tr>
<tr>
<td>ENOLCK</td>
<td>3487</td>
<td>No locks are available.</td>
<td>A system-imposed limit on the number of simultaneous file and record locks was reached, and no more were available at that time.</td>
</tr>
<tr>
<td>ENOTEMPTY</td>
<td>3488</td>
<td>Directory is not empty.</td>
<td>You tried to remove a directory that is not empty. A directory cannot contain objects when it is being removed.</td>
</tr>
<tr>
<td>ENOSYSRSC</td>
<td>3489</td>
<td>System resources are not available.</td>
<td></td>
</tr>
<tr>
<td>ECONVERT</td>
<td>3490</td>
<td>Conversion error.</td>
<td>One or more characters could not be converted from the source CCSID to the target CCSID.</td>
</tr>
<tr>
<td>E2BIG</td>
<td>3491</td>
<td>Argument list is too long.</td>
<td></td>
</tr>
<tr>
<td>EILSEQ</td>
<td>3492</td>
<td>Conversion stopped due to input character that does not belong to the input codeset.</td>
<td></td>
</tr>
<tr>
<td>ETYPE</td>
<td>3493</td>
<td>Object type mismatch.</td>
<td>The type of the object referenced by a descriptor does not match the type specified on the interface.</td>
</tr>
<tr>
<td>EBADDIR</td>
<td>3494</td>
<td>Attempted to reference a directory that was not found or was destroyed.</td>
<td></td>
</tr>
<tr>
<td>EBADOBJ</td>
<td>3495</td>
<td>Attempted to reference an object that was not found, was destroyed, or was damaged.</td>
<td></td>
</tr>
<tr>
<td>EIDXINVAL</td>
<td>3496</td>
<td>Data space index used as a directory is not valid.</td>
<td></td>
</tr>
<tr>
<td>ESOFTDAMAGE</td>
<td>3497</td>
<td>Object has soft damage.</td>
<td>You attempted to use a function that requires you to be enrolled in the system distribution directory and you are not.</td>
</tr>
<tr>
<td>ENOTENROLL</td>
<td>3498</td>
<td>User is not enrolled in system distribution directory.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-----------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EOFLINE</td>
<td>3499</td>
<td>Object is suspended.</td>
<td>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.</td>
</tr>
<tr>
<td>EROOBJ</td>
<td>3500</td>
<td>Object is read-only.</td>
<td>You have attempted to update an object that can be read only.</td>
</tr>
<tr>
<td>EEAHDDSI</td>
<td>3501</td>
<td>Hard damage on extended attribute data space index.</td>
<td></td>
</tr>
<tr>
<td>EEASDDSI</td>
<td>3502</td>
<td>Soft damage on extended attribute data space index.</td>
<td></td>
</tr>
<tr>
<td>EEAHDDS</td>
<td>3503</td>
<td>Hard damage on extended attribute data space.</td>
<td></td>
</tr>
<tr>
<td>EEASDDS</td>
<td>3504</td>
<td>Soft damage on extended attribute data space.</td>
<td></td>
</tr>
<tr>
<td>EEADUPRC</td>
<td>3505</td>
<td>Duplicate extended attribute record.</td>
<td>The read or write of an area conflicts with a lock held by another process.</td>
</tr>
<tr>
<td>ELOCKED</td>
<td>3506</td>
<td>Area being read from or written to is locked.</td>
<td>The read or write of an area conflicts with a lock held by another process.</td>
</tr>
<tr>
<td>EFBIG</td>
<td>3507</td>
<td>Object too large.</td>
<td>The size of the object would exceed the system allowed maximum size.</td>
</tr>
<tr>
<td>EIDRM</td>
<td>3509</td>
<td>The semaphore, shared memory, or message queue identifier is removed from the system.</td>
<td></td>
</tr>
<tr>
<td>ENOMSG</td>
<td>3510</td>
<td>The queue does not contain a message of the desired type and (msgflg logically ANDed with IPC_NOWAIT).</td>
<td></td>
</tr>
<tr>
<td>EFILECVT</td>
<td>3511</td>
<td>File ID conversion of a directory failed.</td>
<td>To recover from this error, run the Reclaim Storage (RCLSTG) command as soon as possible.</td>
</tr>
<tr>
<td>EBADFD</td>
<td>3512</td>
<td>A file ID could not be assigned when linking an object to a directory.</td>
<td>The file ID table is missing or damaged. To recover from this error, run the Reclaim Storage (RCLSTG) command as soon as possible.</td>
</tr>
<tr>
<td>ESTALE</td>
<td>3513</td>
<td>File or object handle rejected by server.</td>
<td></td>
</tr>
<tr>
<td>ESRCH</td>
<td>3515</td>
<td>No such process.</td>
<td></td>
</tr>
<tr>
<td>ENOTSIGINIT</td>
<td>3516</td>
<td>Process is not enabled for signals.</td>
<td>An attempt was made to call a signal function under one of the following conditions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The signal function is being called for a process that is not enabled for asynchronous signals.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The signal function is being called when the system signal controls have not been initialized.</td>
</tr>
<tr>
<td>ECHILD</td>
<td>3517</td>
<td>No child process.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>------------------</td>
<td>-------</td>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EBADH</td>
<td>3520</td>
<td>Handle is not valid.</td>
<td></td>
</tr>
<tr>
<td>ETOOMANYREFS</td>
<td>3523</td>
<td>The operation would have exceeded the maximum number of references allowed for a descriptor.</td>
<td></td>
</tr>
<tr>
<td>ENOTSAFE</td>
<td>3524</td>
<td>Function is not allowed.</td>
<td>Function is not allowed in a job that is running with multiple threads.</td>
</tr>
<tr>
<td>EOVERFLOW</td>
<td>3525</td>
<td>Object is too large to process.</td>
<td>The object’s data size exceeds the limit allowed by this function.</td>
</tr>
<tr>
<td>EJRNNDAMAGE</td>
<td>3526</td>
<td>Journal is damaged.</td>
<td>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.</td>
</tr>
<tr>
<td>EJRNINACTIVE</td>
<td>3527</td>
<td>Journal is inactive.</td>
<td>The journaling state for the journal is *INACTIVE. This error occurs during operations that were attempting to send an entry to the journal.</td>
</tr>
<tr>
<td>EJRNRCVSPC</td>
<td>3528</td>
<td>Journal space or system storage error.</td>
<td>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.</td>
</tr>
<tr>
<td>EJRNRMNT</td>
<td>3529</td>
<td>Journal is remote.</td>
<td>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.</td>
</tr>
<tr>
<td>ENEWJRNRCV</td>
<td>3530</td>
<td>New journal receiver is needed.</td>
<td>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.</td>
</tr>
<tr>
<td>ENEWJRN</td>
<td>3531</td>
<td>New journal is needed.</td>
<td>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.</td>
</tr>
<tr>
<td>EJOURNALED</td>
<td>3532</td>
<td>Object already journaled.</td>
<td>A start journaling operation was attempted on an object that is already being journaled.</td>
</tr>
<tr>
<td>EJRNENTTOOLOG</td>
<td>3533</td>
<td>Entry is too large to send.</td>
<td>The journal entry generated by this operation is too large to send to the journal.</td>
</tr>
<tr>
<td>EDATALINK</td>
<td>3534</td>
<td>Object is a datalink object.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Text</td>
<td>Details</td>
</tr>
<tr>
<td>------------------</td>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENOTAVAIL</td>
<td>3535</td>
<td>Independent Auxiliary Storage Pool (ASP) is not available.</td>
<td>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.</td>
</tr>
<tr>
<td>ENOTTY</td>
<td>3536</td>
<td>I/O control operation is not appropriate.</td>
<td></td>
</tr>
<tr>
<td>EFBIG2</td>
<td>3540</td>
<td>Attempt to write or truncate file past its sort file size limit.</td>
<td></td>
</tr>
<tr>
<td>ETXTBSY</td>
<td>3543</td>
<td>Text file busy.</td>
<td>➤ 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. ⇚</td>
</tr>
<tr>
<td>EASPGPNOTSET</td>
<td>3544</td>
<td>ASP group not set for thread.</td>
<td></td>
</tr>
<tr>
<td>ERESTART</td>
<td>3545</td>
<td>A system call was interrupted and may be restarted.</td>
<td></td>
</tr>
<tr>
<td>ESCANFAILURE</td>
<td>3546</td>
<td>Object had scan failure.</td>
<td>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.</td>
</tr>
</tbody>
</table>
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:
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 information 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.

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:

(C) IBM 2006. Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 1998, 2006. All rights reserved.

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

**Programming Interface Information**

This Application Programming Interfaces (API) 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 Printing
Advanced Peer-to-Peer Networking
AFP
AIX
AS/400
COBOL/400
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI
DRDA
eServer
GDDM
IBM
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
i5/OS
iSeries
Lotus Notes
MVS
Netfinity
Net.Data
NetView
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
PowerPC
PrintManager
Print Services Facility
RISC System/6000
RPG/400
RS/6000
SAA
SecureWay
System/36
System/370
System/38
System/390
VisualAge
WebSphere
xSeries

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, and 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.