IBM Systems - iSeries

i5/OS PASE APIs

Version 5 Release 4

IBM Systems - iSeries

i5/OS PASE APIs

Version 5 Release 4

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 85.

Sixth Edition (February 2006)

This edition applies to version 5, release 4, modification 0 of IBM i5/OS (product number 5722-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

i5/OS PASE APIs

Portable Application Solutions Environment (i5/OS(TM) PASE) is an integrated runtime environment for AIX(R) applications. i5/OS PASE supports the same binary executable format as AIX for PowerPC(R) and a large subset of AIX runtime that allows many AIX applications to run with little or no change.

i5/OS PASE supports direct hardware execution of PowerPC instructions (not an emulator), while providing access to the same i5/OS support used by ILE applications for file systems, sockets, security, and many other system services.

An i5/OS PASE program can be stored in any bytestream file in the i5/OS Integrated File System because it is simply a binary file. i5/OS PASE programs can be created by any compiler and linker that produce executables compatible with AIX for PowerPC.

You must call a system API to run an i5/OS PASE program. The system provides both callable program APIs and ILE procedure APIs to run i5/OS PASE programs. The callable program APIs can be easier to use, but do not offer all the controls available with the ILE procedure APIs.

The functions available to you through i5/OS PASE are:
- "i5/OS PASE Callable Program APIs"
- "i5/OS PASE ILE Procedure APIs" on page 6
- "Runtime Functions For Use by i5/OS PASE Programs" on page 25

See also:
- i5/OS PASE for information about creating i5/OS PASE programs.
- "i5/OS PASE Runtime Libraries" on page 68 for information about i5/OS PASE interfaces that are also supported on AIX.
- "i5/OS PASE Locales" on page 70 for information about i5/OS PASE locales.
- "i5/OS PASE Environment Variables" on page 79 for information about i5/OS PASE environment variables.
- "i5/OS PASE Signal Handling" on page 82 for information about i5/OS PASE signals and how they relate to i5/OS exception messages.

Top | APIs by category

APIs

These are the APIs for this category.

i5/OS PASE Callable Program APIs

The callable program APIs run an i5/OS(TM) PASE program. They are:
- "QP2SHELL() and QP2SHELL2()—Run an i5/OS PASE Shell Program" on page 2 (Run an i5/OS PASE Shell Program) runs an i5/OS PASE program in the job that calls the API.
- "QP2TERM()—Run an i5/OS PASE Terminal Session" on page 5 (Run an i5/OS PASE Terminal Session) runs an interactive terminal session that communicates with an i5/OS PASE program (defaulting to the Korn shell) running in a batch job.

Top | "i5/OS PASE APIs" | APIs by category

QP2SHELL() and QP2SHELL2()—Run an i5/OS PASE Shell Program

Syntax

```
#include <qp2shell.h>

void QP2SHELL(const char    *pathName,
              ...);


#include <qp2shell2.h>

void QP2SHELL2(const char   *pathName,
               ...);
```

Default Public Authority: *USE
Threadsafe: No

Programs QP2SHELL and QP2SHELL2 run an i5/OS Portable Application Solutions Environment (i5/OS PASE) program in the job where the API is called. They load the i5/OS PASE program and any necessary shared libraries and then transfer control to the program. QP2SHELL runs in a new ILE activation group, while QP2SHELL2 runs in the caller's activation group. Control returns to the caller when the i5/OS PASE program either exits, terminates due to a signal, or returns without exiting.

Parameters

pathName
(Input) Pointer to a null-terminated character string that identifies the stream file in the Integrated File System that contains the i5/OS PASE program to run. The pathName string may include an absolute or relative path qualifier in addition to the stream file name. Relative path names are resolved using the current working directory.

If the base name part of the pathName value (excluding any prefix path qualifier) begins with a hyphen (-), QP2SHELL and QP2SHELL2 strip the hyphen when locating the bytestream file, but pass the full string (with the hyphen) to the i5/OS PASE program as the program name. Standard i5/OS PASE shell programs (including sh and ksh) run as login shells when called with a hyphen as the first character of the program name. Login shells look for a profile file and run it automatically when the shell starts.

argument strings
(Input) Optional pointers to null-terminated character strings that are passed to the i5/OS PASE program as arguments. The system copies argument strings into i5/OS PASE memory and converts them from the job default CCSID to the CCSID specified by ILE environment variable QIBM_PASE_CCSID.

Note: When calling QP2SHELL or QP2SHELL2 from CL, be sure to quote any argument string that could be interpreted as a numeric value. CL converts unquoted numeric arguments to decimal or floating-point format, which does not match the assumption made by these APIs and i5/OS PASE programs that all arguments are null-terminated character strings.

Authorities

Object Referred to	Authority Required
Each directory in the path to the i5/OS PASE program and shared libraries	*X
i5/OS PASE program (not a shell script) in a local file system	*X
i5/OS PASE program in a remote file system or shell script	*RX

Object Referred to	Authority Required
i5/OS PASE shared library	*R

Return Value

QP2SHELL and QP2SHELL2 return no function result. Escape messages are sent to report errors.

Error Messages

Some of the more common error messages sent by QP2SHELL and QP2SHELL2 are:

Message ID	Error Message Text
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFB9C0 E	Error loading program &1. See previous messages.
CPFB9C1 E	System support for i5/OS Portable Application Solutions Environment not available.
CPFB9C2 E	Hardware support for i5/OS Portable Application Solutions Environment not available.
CPFB9C3 E	i5/OS PASE CCSID and job default CCSID are not compatible.
CPFB9C5 E	i5/OS PASE program name required by QP2SHELL.
CPFB9C6 E	i5/OS PASE ended for signal &1, error code &2.
CPFB9C7 E	i5/OS PASE already running in this job.
CPFB9C8 E	File descriptors 0, 1, and 2 must be open to run the i5/OS PASE program.

Usage Notes

1. QP2SHELL and QP2SHELL2 provide callable program interfaces to ILE procedure Qp2RunPase. See "Qp2RunPase()—Run an i5/OS PASE Program" on page 20 for details about running an i5/OS PASE program.
2. QP2SHELL and QP2SHELL2 set the ILE pthread cancel state and cancel type to default values (PTHREAD_CANCEL_ENABLE and PTHREAD_CANCEL_DEFERRED) before running the i5/OS PASE program. This is done to avoid unexpected behavior for the i5/OS PASE program if the job changed ILE pthread attributes before calling the API.
3. » QP2SHELL, QP2SHELL2, and system processing for the i5/OS PASE fork function normally set up handlers for most ILE signals (replacing any prior handlers). QP2SHELL always restores original ILE signal handlers before returning to the caller. QP2SHELL2 restores original ILE signal handlers before returning if the i5/OS PASE program exits, but if the i5/OS PASE program returns without exiting, original ILE signal handlers are not restored until the system destroys the activation group that called QP2SHELL2.

   You can control how these interfaces handle signals by setting ILE environment variable *QIBM_PASE_MAP_SIGNALS* to one of these values:

Y	Call Qp2SignalPase for any ILE signal (other than SIGCHLD) that corresponds to an i5/OS PASE signal. The pending i5/OS PASE signal that may result is not delivered until control transfers to the i5/OS PASE program in some thread that has not blocked the signal. This is the default behavior if QIBM_PASE_MAP_SIGNALS is not set.
I	Call Qp2SignalPase for any ILE signal (other than SIGCHLD) that corresponds to an i5/OS PASE signal, and attempt to deliver pending signals immediately by using Qp2CallPase to call an i5/OS PASE function (in the same thread). A signal may remain pending if no thread running i5/OS PASE code has the signal unblocked.
N	Do not map any ILE signals to i5/OS PASE. No ILE signal handlers are changed. i5/OS PASE runtime functions that rely on system-generated signals (such as asynchronous I/O use of SIGIO and SIGURG) may not work correctly.

«

4. To avoid unpredictable results, do not not change ILE environment variables QIBM_USE_DESCRIPTOR_STDIO or QIBM_PASE_DESCRIPTOR_STDIO in a job in which an i5/OS PASE program is running.
5. QP2SHELL and QP2SHELL2 initialize i5/OS PASE environment variables with a modified copy of the entire ILE environment. An i5/OS PASE environment variable is initialized for every ILE environment variable, but the initial value of any i5/OS PASE variable (except those whose name begins with "PASE_") can be overridden by the value of an ILE environment variable with a name that concatenates the prefix PASE_ with the original variable name. This processing avoids some interference between i5/OS PASE runtime and ILE runtime when they require different values for the same environment variable (for example, LANG).
6. For a login shell (only), QP2SHELL and QP2SHELL2 set ILE environment variable PASE_SHELL to the path name of the i5/OS PASE shell program.
7. QP2SHELL and QP2SHELL2 initialize any of the following ILE environment variables that are not already set, with default values as shown:

HOME	If HOME is not already set, QP2SHELL and QP2SHELL2 set it to the home directory path specified in the user profile identified by the LOGIN variable. If the job is not currently authorized to the LOGIN user profile, the HOME environment variable is set to a null string.
LOGIN	If LOGIN is not already set, QP2SHELL and QP2SHELL set it to the middle qualifier of the job name. For an interactive job, this is the name of the user who did a signon to start the job.
PASE_PATH	(Default: "/QOpenSys/usr/bin:/usr/ccs/bin:/QOpenSys/usr/bin/X11:/usr/sbin:.:/usr/bin") Initial value for the i5/OS PASE PATH environment variable.
PASE_LANG and QIBM_PASE_CCSID	Initial value for the i5/OS PASE LANG environment variable and what coded character set identifier (CCSID) the i5/OS PASE program will use. QP2SHELL and QP2SHELL2 set both these ILE environment variables if either or both is absent. The default values are function of the current LANGID and CNTRYID attributes of the job, but the system will use PASE_LANG=POSIX and QIBM_PASE_CCSID=819 if it does not recognize the LANGID and CNTRYID pair. The i5/OS PASE LANG environment variable controls the default locale for an i5/OS PASE program. See "i5/OS PASE Locales" on page 70 to determine what locales are supported by i5/OS PASE.
PASE_LOCPATH	(Default: "/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat") Initial value for the i5/OS PASE LOCPATH environment variable.
PASE_LC__FASTMSG	(Default: "true") Initial value for the i5/OS PASE LC__FASTMSG environment variable.
PASE_TZ	(Default: based on the i5/OS job TIMZON attribute) Initial value for the i5/OS PASE TZ environment variable. If no timezone information is provided in environment variable TZ, the i5/OS PASE program sees UTC (Universal Standard Time) as local time. You can set ILE environment variable PASE_TZ at the system level to provide a default timezone other than the one determined from the job TIMZON attribute. For example, this CL command sets the default timezone to US Central time: `ADDENVVAR ENVVAR(PASE_TZ) VALUE('CST6CDT') LEVEL(*SYS)`
QIBM_IFS_OPEN_MAX	(Default: "66000") Maximum number of Integrated File System open file descriptors desired in the job. QP2SHELL and QP2SHELL call the DosSetRelMaxFH API to set the maximum number of file descriptors to the value in this ILE environment variable, and updates the environment variable to reflect the actual limit (in case the requested limit is not currently allowed). Any change to the maximum number of file descriptors persists after the API returns. i5/OS PASE programs assume the ability to open 65 534 files and the system requires an open file for each i5/OS PASE executable it loads, so the default of 66 000 files accomodates a maximally large i5/OS PASE program with a fairly large number of loaded executables.

Related Information

- "Qp2CallPase()—Call an i5/OS PASE Procedure" on page 8
- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20
- "Qp2SignalPase()—Post an i5/OS PASE Signal" on page 24
- "QP2TERM()—Run an i5/OS PASE Terminal Session"

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

QP2TERM()—Run an i5/OS PASE Terminal Session

Syntax

```
#include <qp2term.h>

void QP2TERM(...);
```

Default Public Authority: *USE
Threadsafe: No

The QP2TERM() program runs an interactive terminal session that starts a batch job to run an i5/OS Portable Application Solutions Environment (i5/OS PASE) program. This program uses the workstation display in the interactive to present output and accept input for files stdin, stdout, and stderr in the batch job.

Parameters

argument strings

(Input) Optional pointers to null-terminated character strings that specify the path name of the i5/OS PASE program to run and any argument strings to pass to the program. If no parameters are specified, QP2TERM runs the default i5/OS PASE shell as an interactive login shell. The default i5/OS PASE shell is an implementation of the Korn shell, with path name /QOpenSys/usr/bin/sh.

Note: When calling QP2TERM from CL, be sure to quote any argument string that could be interpreted as a numeric value. CL converts unquoted numeric arguments to decimal or floating-point format, which does not match the assumption made by QP2TERM and i5/OS PASE programs that all arguments are null-terminated character strings.

Authorities

Object Referred to	Authority Required
Each directory in the path to the i5/OS PASE program and shared libraries	*X
i5/OS PASE program (not a shell script) in a local file system	*X
i5/OS PASE program in a remote file system or shell script	*RX
i5/OS PASE shared library	*R

Return Value

QP2TERM returns no function result. Escape messages are sent to report errors.

Error Messages

Message ID	Error Message Text
CPFB9C4 E	Error running i5/OS PASE terminal session, reason code &1, errno &2.
CPFB9C9 E	Terminal session already in use.
CPFB9CA E	Batch job ended in error.

Usage Notes

1. QP2TERM uses the Qp0zStartTerminal API to manage the interactive display and start a batch job. The batch job copies most attributes of the interactive job and calls program QP2SHELL to run the i5/OS PASE program. See "QP2SHELL() and QP2SHELL2()—Run an i5/OS PASE Shell Program" on page 2 for details about running an i5/OS PASE shell program.
2. QP2TERM copies all ILE environment variables from the interactive job to the batch job before starting the batch job, except the following ILE environment variables, which are set or replaced in the batch job. These changes affect the batch job only. They do not modify the environment in the job that called QP2TERM.

COLUMNS	If COLUMNS is not already set, QP2TERM sets it to the number of columns available for program output on the interactive display.
ROWS	If ROWS is not already set, QP2TERM sets it to the number of rows available for program output on the interactive display.
QIBM_USE_DESCRIPTOR_STDIO=I	QP2TERM sets QIBM_USE_DESCRIPTOR_STDIO to ensure that files stdin, stdout, and stderr use Integrated File System descriptors 0, 1, and 2. The terminal session manager attaches open pipes to these file descriptors in the batch job.
QIBM_PASE_DESCRIPTOR_STDIO=T	QP2TERM sets QIBM_PASE_DESCRIPTOR_STDIO to ensure that i5/OS PASE runtime does ASCII/EBCDIC text conversion for data that the i5/OS PASE program reads or writes to files stdin, stdout, and stderr.

Related Information

- Qp0zStartTerminal()—Start a Terminal Session
- "QP2SHELL() and QP2SHELL2()—Run an i5/OS PASE Shell Program" on page 2

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

i5/OS PASE ILE Procedure APIs

The ILE procedure APIs run an i5/OS(TM) PASE program and allow ILE programs to communicate with an i5/OS PASE program that is already running in the same job.

The i5/OS PASE ILE Procedure APIs are:

- "Qp2dlclose()—Close a Dynamically Loaded i5/OS PASE Module" on page 7 (Close a Dynamically Loaded i5/OS PASE Module) closes and unloads an i5/OS PASE module previously opened by the Qp2dlopen API (or the i5/OS PASE dlopen function).
- "Qp2CallPase()—Call an i5/OS PASE Procedure" on page 8 (Call an i5/OS PASE Procedure) calls a procedure in an i5/OS PASE program that is already running in the job that calls the API.

- “Qp2dlerror()—Retrieve i5/OS PASE Dynamic Load Error Information” on page 11 (Retrieve i5/OS PASE Dynamic Load Error Information) returns a pointer to a string that provides error information for the most recent dynamic load function (Qp2dlopen, Qp2dlsym, or Qp2dlclose API).
- “Qp2dlopen()—Dynamically Load an i5/OS PASE Module” on page 12 (Dynamically Load an i5/OS PASE Module) dynamically loads an i5/OS PASE module by calling the i5/OS PASE dlopen() function.
- “Qp2dlsym()—Find an Exported i5/OS PASE Symbol” on page 14 (Find an Exported i5/OS PASE Symbol) finds an exported i5/OS PASE symbol by calling the i5/OS PASE dlsym() function.
- “Qp2EndPase()—End an i5/OS PASE Program” on page 15 (End an i5/OS PASE Program) ends any i5/OS PASE program currently running in the job.
- “Qp2errnop()—Retrieve i5/OS PASE errno Pointer” on page 15 (Retrieve i5/OS PASE errno Pointer) returns a pointer to the i5/OS PASE errno variable for the current thread.
- “Qp2free()—Free i5/OS PASE Heap Memory” on page 16 (Free i5/OS PASE Heap Memory) frees an i5/OS PASE heap memory allocation by calling the i5/OS PASE free() function.
- “Qp2jobCCSID()—Retrieve Job CCSID for i5/OS PASE” on page 17 (Retrieve Job CCSID for i5/OS PASE) returns the job default CCSID from the last time the i5/OS PASE CCSID was set.
- “Qp2malloc()—Allocate i5/OS PASE Heap Memory” on page 17 (Allocate i5/OS PASE Heap Memory) allocates memory from the i5/OS PASE heap by calling the i5/OS PASE malloc() function.
- “Qp2paseCCSID()—Retrieve i5/OS PASE CCSID” on page 18 (Retrieve i5/OS PASE CCSID) returns the i5/OS PASE CCSID from the last time the i5/OS PASE CCSID was set.
- “Qp2ptrsize()—Retrieve i5/OS PASE Pointer Size” on page 19 (Retrieve i5/OS PASE Pointer Size) returns the pointer size, in bytes, for the i5/OS Application Solutions Environment (i5/OS PASE) program currently running in the job.
- “Qp2RunPase()—Run an i5/OS PASE Program” on page 20 (Run an i5/OS PASE Program) runs an i5/OS PASE program in the job that calls the API.
- “Qp2SignalPase()—Post an i5/OS PASE Signal” on page 24 (Post an i5/OS PASE Signal) posts an i5/OS PASE signal to an i5/OS PASE program that is already running in the job that calls the API.

Top | “i5/OS PASE APIs,” on page 1 | APIs by category

Qp2dlclose()—Close a Dynamically Loaded i5/OS PASE Module

Syntax

```
#include <qp2user.h>

int Qp2dlclose(QP2_ptr64_t  id);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: Yes

Qp2dlclose() closes and unloads an i5/OS PASE module previously opened by API Qp2dlopen (or the i5/OS PASE dlopen function).

Parameters

id (Input) Specifies a value returned by API Qp2dlopen (or the i5/OS PASE dlopen function) that specifies what module is closed and unloaded.

Authorities

None.

Return Value

The function result is zero for normal completion, or -1 with an error indicated in ILE **errno** or i5/OS PASE **errno** (if ILE **errno** is zero). You can also call API Qp2dlerror for more information about any error.

Related Information

- i5/OS PASE dlclose()—See AIX documentation
- "Qp2dlerror()—Retrieve i5/OS PASE Dynamic Load Error Information" on page 11
- "Qp2errnop()—Retrieve i5/OS PASE errno Pointer" on page 15
- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20

API introduced: V5R2

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Qp2CallPase()—Call an i5/OS PASE Procedure

Syntax

```
#include <qp2user.h>

int Qp2CallPase(const void             *target,
                const void             *arglist,
                const QP2_arg_type_t   *signature,
                QP2_result_type_t      result_type,
                void                   *buf);


int Qp2CallPase2(const void            *target,
                 const void            *arglist,
                 const QP2_arg_type_t  *signature,
                 QP2_result_type_t     result_type,
                 void                  *buf,
                 short                 bufLenIn);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: Yes

The Qp2CallPase() and Qp2CallPase2 functions call a procedure in an i5/OS Portable Application Solutions Environment (i5/OS PASE) program in a job that is already running the i5/OS PASE program.

Parameters

target (Input) Pointer to a function descriptor for the procedure (in the i5/OS PASE program) to call. The format and contents of a function descriptor are specified by the PowerPC Application Binary Interface (ABI) for AIX. A function descriptor contains three i5/OS PASE addresses (not MI pointers) that point to the executable instructions, table of contents (TOC), and environment for the target procedure.

arglist (Input) Pointer to the argument list for the i5/OS PASE procedure. The format and contents of a PASE argument list generally are specified by the PowerPC ABI for AIX. The specific argument list structure for the i5/OS PASE procedure identified by the target parameter is determined by the list of argument data types specified by the signature parameter.

signature

(Input) Pointer to an array of values that specify the sequence and type of arguments passed to the i5/OS PASE procedure. Each element in the array is either a special value defined in header file qp2user.h or a positive number that is the length in bytes of a structure or union argument passed by value. The last value in the array must be QP2_ARG_END. Header file qp2user.h defines the following constants for the data types supported as arguments for an i5/OS PASE procedure:

QP2_ARG_END (0)	The end of the list of argument type values.
QP2_ARG_WORD (-1)	A 4-byte signed or unsigned integer, or a structure or union no longer than four bytes. This value is allowed only when calling a procedure in a 32-bit i5/OS PASE program.
QP2_ARG_DWORD (-2)	An 8-byte signed or unsigned integer, or a structure or union no longer than eight bytes. This value is allowed only when calling a procedure in a 64-bit i5/OS PASE program.
QP2_ARG_FLOAT32 (-3)	A 4-byte floating point number.
QP2_ARG_FLOAT64 (-4)	An 8-byte floating point number.
QP2_ARG_PTR32 (-5)	A 4-byte pointer. The value in the arglist buffer is passed unchanged unless its high-order bits (excluding the lower 16 bits) match the corresponding part of constant QP2_ARG_PTR_TOSTACK (0x0fff0000). In that case, the arglist value is changed to the memory address used for a copy of the buf area plus an offset in the lower 16 bits of the arglist value, and the updated value is passed to the i5/OS PASE procedure. QP2_ARG_PTR32 is allowed only when calling a procedure in a 32-bit i5/OS PASE program.
QP2_ARG_PTR64 (-6)	An 8-byte pointer. The value in the arglist buffer is passed unchanged unless its high-order bits (excluding the lower 16 bits) match the corresponding part of constant QP2_ARG_PTR_TOSTACK (0x000000000fff0000). In that case, the arglist value is changed to the memory address used for a copy of the buf area plus an offset in the lower 16 bits of the arglist value, and the updated value is passed to the i5/OS PASE procedure. QP2_ARG_PTR64 is allowed only when calling a procedure in a 64-bit i5/OS PASE program.

result_type

(Input) The data type of the function result returned by the i5/OS PASE procedure. Result_type is either a special value defined in header file qp2user.h or a positive number that is the length in bytes of by-address result data copied from the i5/OS PASE stack to the buf area after the i5/OS PASE procedure returns. Header file qp2user.h defines the following constants for function result data types:

QP2_RESULT_VOID (0)	No function result returned.
QP2_RESULT_WORD (-1)	A 4-byte signed or unsigned integer, or a structure or union no longer than four bytes. This value is allowed only when calling a procedure in a 32-bit i5/OS PASE program.
QP2_RESULT_DWORD (-2)	An 8-byte signed or unsigned integer, or a structure or union no longer than eight bytes returned by a procedure in a 64-bit i5/OS PASE program.
QP2_RESULT_FLOAT64 (-4)	An 8-byte floating point number.
QP2_RESULT_PTR32 (-5)	A 4-byte pointer. A pointer result from the i5/OS PASE procedure is returned unchanged. This value is allowed only when calling a procedure in a 32-bit i5/OS PASE program.
QP2_RESULT_PTR64 (-6)	An 8-byte pointer. A pointer result from the i5/OS PASE procedure is returned unchanged. This value is allowed only when calling a procedure in a 64-bit i5/OS PASE program.

buf (Input/Output) Pointer to a buffer that contains by-address argument data and the function result. buf is ignored if result_type is QP2_RESULT_VOID and bufLenIn is either zero or omitted (for Qp2CallPase).

bufLenIn
(Input) Length of by-address argument input data. A positive number specifies the number of bytes copied from the buf area to the i5/OS PASE stack before the i5/OS PASE procedure is called.

Authorities

None.

Return Value

The function result is an integer that indicates whether the i5/OS PASE function was called successfully. Header file qp2user.h defines the following constants for the return code from Qp2CallPase and Qp2CallPase2:

QP2CALLPASE_NORMAL (0)	The i5/OS PASE procedure ran to completion and its function result (if any) was stored in the location identified by the buf parameter.
QP2CALLPASE_RESULT_ERROR (1)	The i5/OS PASE procedure ran to completion, but its function result could not be stored at the location identified by the buf parameter. buf may be a null pointer value, or the space addressed by buf may be damaged or destroyed.
QP2CALLPASE_ENVIRON_ERROR (2)	The operation is not allowed because no i5/OS PASE program is running in the job, or the thread that called Qp2CallPase or Qp2CallPase2 was neither the initial i5/OS PASE thread nor a thread created using i5/OS PASE pthread interfaces.
QP2CALLPASE_ARG_ERROR (4)	One or more values in the signature array are not valid.
QP2CALLPASE_TERMINATING (6)	The i5/OS PASE program is terminating. No function result was returned. The i5/OS PASE program may have run the exit function, or a signal might have caused the program to terminate.
QP2CALLPASE_RETURN_NOEXIT (7)	The i5/OS PASE program returned without exiting by calling the i5/OS PASE **_RETURN** function. No function result was returned.

Usage Notes

1. Qp2CallPase and Qp2CallPase2 are supported only when an i5/OS PASE program is currently running in the job. This means that Qp2RunPase must be running actively in the job, or the job must be a fork child process.
2. You can run Qp2CallPase and Qp2CallPase2 only in the initial thread that started the i5/OS PASE program or in a thread created using i5/OS PASE pthread interfaces, unless i5/OS PASE environment variable PASE_THREAD_ATTACH was set to Y when a thread-enabled i5/OS PASE program was started.
3. Once an ILE thread has attached to i5/OS PASE (by calling an i5/OS PASE procedure), that thread is subject to asynchronous interruption for i5/OS PASE functions such as signal handling and thread cancellation. In particular, the thread will be canceled as part of ending the i5/OS PASE program (when **exit** runs or i5/OS PASE processing terminates for a signal).
4. An i5/OS PASE procedure called by Qp2CallPase or Qp2CallPase2 must return to its caller. Unpredictable results occur if the i5/OS PASE procedure attempts to longjmp to an older call or if it performs an operation that terminates the thread or process (such as calling the exit function). If a signal handler is on the i5/OS PASE stack when Qp2CallPase or Qp2CallPase2 is called, the called i5/OS PASE procedure must also honor restrictions on runtime functions allowed in signal handlers (see AIX signal handling documentation for details).
5. A pointer to any function in an i5/OS PASE program is really a pointer to a function descriptor for the procedure. An i5/OS PASE program can easily provide a function descriptor to ILE code by passing an i5/OS PASE function pointer value converted to an ILE memory address. The conversion can be done using the _SETSPP function or the ARG_MEMPTR argument type on the _ILECALLX or _ILECALL function.

6. Qp2CallPase and Qp2CallPase2 support arguments and results passed by-address through the use of QP2_ARG_PTR32 or QP2_ARG_PTR64 values in the signature array and positive numbers for the result_type and/or bufLenIn arguments.
7. If the buf area is 16-byte aligned, any tagged ILE pointers are preserved in by-address (input) argument data copied from the buf area to i5/OS PASE memory, and in by-address result data copied from i5/OS PASE memory to the buf area.
8. A structure or union function result returned by-value that is short enough to fit into a register must be handled as QP2_RESULT_WORD for a 32-bit i5/OS PASE program or as QP2_RESULT_DWORD for a 64-bit i5/OS PASE program. Longer structure or union function results returned by-value are actually returned by-address through a buffer pointer passed as the first (hidden) argument to the i5/OS PASE procedure.
9. You may need to limit result_type and bufLenIn to avoid overrunning the end of the i5/OS PASE stack. Arguments and results that are too large for the stack can be passed by-address using argument pointers to i5/OS PASE heap storage.
10. The PowerPC ABI for AIX requires 4-byte alignment for each argument passed to a procedure in a 32-bit program, and 8-byte alignment for each argument passed to a procedure in a 64-bit program. Qp2CallPase and Qp2CallPase2 assume the caller provides an arglist data structure that provides this alignment, including any necessary pad bytes following a structure or union argument and following a QP2_ARG_FLOAT32 argument passed to a 64-bit i5/OS PASE program. The arglist structure also needs to store any 64-bit integer or floating point argument on a 4-byte boundary when the target procedure is in a 32-bit i5/OS PASE program (rather than the 8-byte boundary used as the default for these types in ILE C and C++ compilers).

Related Information

- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20—Run an i5/OS PASE Program

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Qp2dlerror()—Retrieve i5/OS PASE Dynamic Load Error Information

Syntax

```
#include <qp2user.h>

char* Qp2dlerror(void);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: No

Qp2dlerror() returns a pointer to a string that provides error information for the most recent dynamic load function (API Qp2dlopen, Qp2dlsym, or Qp2dlclose).

Parameters

None.

Authorities

None.

Return Value

The function result is a pointer to a null-terminated character string (in the job default CCSID). A null pointer is returned if no error occurred during the most recent dynamic load operation. Once Qp2dlerror is called, subsequent calls without an intervening dynamic load error also return a null pointer.

The ILE **errno** is set and a null pointer is returned for any internal processing error (such as an error converting the string from the i5/OS PASE CCSID to the job default CCSID).

Usage Notes

1. Qp2dlerror is not threadsafe because it may call an i5/OS PASE function that is not threadsafe (dlerror) and uses a buffer in static storage for the error string that is also updated by other dynamic load functions (APIs Qp2dlopen, Qp2dlsym, and Qp2dlclose). Applications may need to serialize use of dynamic load functions and copy the error information string to preserve its contents.

Related Information

- i5/OS PASE dlerror()—See AIX documentation
- “Qp2errnop()—Retrieve i5/OS PASE errno Pointer” on page 15
- “Qp2RunPase()—Run an i5/OS PASE Program” on page 20

API introduced: V5R2

Top | “i5/OS PASE APIs,” on page 1 | APIs by category

Qp2dlopen()—Dynamically Load an i5/OS PASE Module

Syntax

```
#include <qp2user.h>

QP2_ptr64_t Qp2dlopen(const char  *path,
                      int         flags,
                      int         ccsid);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: Yes

Qp2dlopen() dynamically loads an i5/OS PASE module by calling the i5/OS PASE dlopen() function.

Parameters

path (Input) A pointer to a null-terminated string that identifies the stream file in the Integrated File System that contains the i5/OS PASE module to load. This API copies the input path string and converts the copy from the CCSID specified by the **ccsid** argument to the current i5/OS PASE CCSID (required by the i5/OS PASE dlopen function).

If the input path pointer is null, the function result is a value for the main application that lets you find symbols in the i5/OS PASE process global name space, which includes all symbols exported by the i5/OS PASE program and shared executables except those loaded by i5/OS PASE dlopen using option RTLD_LOCAL.

flags (Input) Flags passed to the i5/OS PASE dlopen function to control its behavior. These constants, declared in qp2user.h, match constants in AIX header dlfcn.h (without the leading prefix, QP2_) and can be **OR**ed together for the flags argument:

QP2_RTLD_NOW (0x00000002)	Load all dependents of the module being loaded and resolve all symbols. Either QP2_RTLD_NOW or QP2_RTLD_LAZY must be specified.
QP2_RTLD_LAZY (0x00000004)	Allow the system to defer loading dependent modules. Either QP2_RTLD_NOW or QP2_RTLD_LAZY must be specified.
QP2_RTLD_GLOBAL (0x00010000)	Load the module into the global name space. Exported symbols in the module will be visible in the main application and will be used when resolving symbols used by other i5/OS PASE dlopen calls.
QP2_RTLD_LOCAL (0x00080000)	Load the module into a local name space. This option is the default when neither QP2_RTLD_GLOBAL nor QP2_RTLD_LOCAL is specified. It prevents symbols in the module being loaded from being used when resolving symbols used by other dlopen calls.
QP2_RTLD_MEMBER (0x00040000)	Specifies that the *path* argument string may contain the name of a member in an archive (shared library).
QP2_RTLD_NOAUTODEFER (0x00020000)	Prevent deferred imports in the module being loaded from being automatically resolved by subsequent loads.

ccsid (Input) Specifies the CCSID for the input *path* argument string. Zero means the path is in the (EBCDIC) job default CCSID.

Authorities

Object Referred to	Authority Required
Each directory in the path to the i5/OS PASE module	*X
i5/OS PASE module	*R

Return Value

Sucessful completion returns a non-zero function result that can be used to call APIs Qp2dlsym and Qp2dlclose (and also i5/OS PASE functions dlsym and dlclose). Resources allocated for the function result are not freed until the i5/OS PASE program ends or the value is passed to API Qp2dlclose (or i5/OS PASE dlclose).

A zero function result indicates an error. The caller can check ILE **errno** or i5/OS PASE **errno** (if ILE **errno** is zero), or call the Qp2dlerror API for more information about the error.

Related Information

- i5/OS PASE dlopen()—See AIX documentation
- “Qp2dlerror()—Retrieve i5/OS PASE Dynamic Load Error Information” on page 11
- “Qp2errnop()—Retrieve i5/OS PASE errno Pointer” on page 15
- “Qp2RunPase()—Run an i5/OS PASE Program” on page 20

API introduced: V5R2

Top | “i5/OS PASE APIs,” on page 1 | APIs by category

Qp2dlsym()—Find an Exported i5/OS PASE Symbol

Syntax

```
#include <qp2user.h>

void* Qp2dlsym(QP2_ptr64_t  id
               const char   *name,
               int          ccsid,
               QP2_ptr64_t  *sym_pase);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: Yes

Qp2dlsym() finds an exported i5/OS PASE symbol by calling the i5/OS PASE dlsym() function.

Parameters

id (Input) Specifies a value returned by API Qp2dlopen (or the i5/OS PASE dlopen function) that controls what modules are searched for the exported symbol.

name (Input) A pointer to a null-terminated string that contains the symbol name. This API copies the input name string and converts the copy from the CCSID specified by the **ccsid** argument to the current i5/OS PASE CCSID (required by the i5/OS PASE dlsym function).

ccsid (Input) Specifies the CCSID for the input *name* argument string. Zero means the symbol name is in the (EBCDIC) job default CCSID.

sym_pase
(Input) A pointer to a buffer, used to return the i5/OS PASE address of the exported symbol. The return value is always 64-bits, even for a 32-bit i5/OS PASE program. sym_pase can be null if the caller does not need the i5/OS PASE address of the symbol.

Authorities

None.

Return Value

The function result is a pointer to the specified symbol, or a null pointer if the symbol could not be resolved. A buffer addressed by the sym_pase argument is unchanged if the symbol could not be resolved. The caller can check ILE **errno** or i5/OS PASE **errno** (if ILE **errno** is zero), or call the Qp2dlerror API for more information about any error.

Related Information

- i5/OS PASE dlsym()—See AIX documentation
- “Qp2dlerror()—Retrieve i5/OS PASE Dynamic Load Error Information” on page 11
- “Qp2errnop()—Retrieve i5/OS PASE errno Pointer” on page 15
- “Qp2RunPase()—Run an i5/OS PASE Program” on page 20

API introduced: V5R2

Top | “i5/OS PASE APIs,” on page 1 | APIs by category

Qp2EndPase()—End an i5/OS PASE Program

Syntax

```
#include <qp2user.h>

int Qp2EndPase(void);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: No

The Qp2EndPase() function ends any i5/OS PASE program currently running in the job.

Parameters

None.

Authorities

None.

Return Value

The function result is nonzero if an error is detected attempting to end the i5/OS PASE program.

Usage Notes

1. Qp2EndPase is normally used to end an i5/OS PASE program that ran the **_RETURN** i5/OS PASE runtime function (to return without exiting). Such a program remains active (even if it exits or terminates due to an i5/OS PASE signal) until either Qp2EndPase is called or the ILE activation group that called the Qp2RunPase API exits. i5/OS PASE programs that do not use **_RETURN** are ended automatically before control returns from the Qp2RunPase API.
2. Qp2EndPase returns without error when no i5/OS PASE program is running in the job.
3. » Undefined behavior results if Qp2EndPase is called while the Qp2RunPase API is running (in the same job), or if the job attempts to use the i5/OS PASE program (without restarting it) after Qp2EndPase is called.«

Related Information

- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20
- "_RETURN()—Return Without Exiting i5/OS PASE" on page 59

API introduced: V5R2

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Qp2errnop()—Retrieve i5/OS PASE errno Pointer

Syntax

```
#include <qp2user.h>

int* Qp2errnop(void);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: Yes

Qp2errnop() returns a pointer to the i5/OS PASE **errno** variable for the current thread.

Parameters

None.

Authorities

None.

Return Value

The function result is a pointer to the i5/OS PASE **errno** variable for the current thread, or a null pointer if **errno** location is not available (such as when no i5/OS PASE program is running in the job).

Related Information

- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20

API introduced: V5R2

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Qp2free()—Free i5/OS PASE Heap Memory

Syntax

```
#include <qp2user.h>

int Qp2free(void *mem);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: Yes

Qp2free() frees an i5/OS PASE heap memory allocation by calling the i5/OS PASE free() function.

Parameters

mem (Input) A pointer to the start of the i5/OS PASE memory allocation to be freed.

Authorities

None.

Return Value

The function result is zero for normal completion, or -1 with an error indicated in ILE **errno** that is ususally one of the following:

EPERM	An error occurred attempting to call an i5/OS PASE function.
ETERM	i5/OS PASE is terminating.

Related Information

- i5/OS PASE free()—See AIX documentation
- "Qp2errnop()—Retrieve i5/OS PASE errno Pointer" on page 15

- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20

API introduced: V5R2

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Qp2jobCCSID()—Retrieve Job CCSID for i5/OS PASE

Syntax

```
#include <qp2user.h>             /* for ILE programs */
#include <as400_protos.h>        /* for i5/OS PASE programs */

int Qp2jobCCSID(void);
```

Service Program Name: QP2USER (for ILE programs)
i5/OS PASE Library: libc.a (for i5/OS PASE programs)
Default Public Authority: *USE
Threadsafe: Yes

Note: This function can be used in either an ILE program or an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

Qp2jobCCSID() returns the job default CCSID (coded character set identifier) from the last time the i5/OS PASE CCSID was set. The i5/OS PASE CCSID is set when an i5/OS PASE program starts, and can be changed by the i5/OS PASE runtime function _SETCCSID.

Parameters

None.

Authorities

None.

Return Value

The function result is a coded character set identifier (CCSID), or 0 if i5/OS PASE CCSID information is not available (such as when no i5/OS PASE program is running in the job).

Related Information

- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20
- "Qp2paseCCSID()—Retrieve i5/OS PASE CCSID" on page 18

API introduced: V5R2

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Qp2malloc()—Allocate i5/OS PASE Heap Memory

Syntax

```
#include <qp2user.h>

void* Qp2malloc(QP2_dword_t size,
                QP2_ptr64_t *mem_pase);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: Yes

Qp2malloc() allocates memory from the i5/OS PASE heap by calling the i5/OS PASE malloc() function.

Parameters

size (Input) The size, in bytes, of the desired memory allocation.

mem_pase
(Input) A pointer to a buffer, used to return the i5/OS PASE address of the allocated memory. The return value is always 64-bits, even for a 32-bit i5/OS PASE program. mem_pase can be null if the caller does not need the i5/OS PASE address of the memory allocation.

Authorities

None.

Return Value

The function result is a pointer to the i5/OS PASE heap memory allocation, or a null pointer if no memory was allocated. A buffer addressed by the mem_pase argument is unchanged if no memory was allocated.

Related Information

- i5/OS PASE malloc()—See AIX documentation
- "Qp2errnop()—Retrieve i5/OS PASE errno Pointer" on page 15
- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20

API introduced: V5R2

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Qp2paseCCSID()—Retrieve i5/OS PASE CCSID

Syntax

```
#include <qp2user.h>            /* for ILE programs */
#include <as400_protos.h>       /* for i5/OS PASE programs */

int Qp2paseCCSID(void);
```

Service Program Name: QP2USER (for ILE programs)
i5/OS PASE Library: libc.a (for i5/OS PASE programs)
Default Public Authority: *USE
Threadsafe: Yes

Note: This function can be used in either an ILE program or an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

Qp2paseCCSID() returns the i5/OS PASE CCSID (coded character set identifier) from the last time the i5/OS PASE CCSID was set. The i5/OS PASE CCSID is set when an i5/OS PASE program starts, and can be changed by the i5/OS PASE runtime function _SETCCSID.

Parameters

None.

Authorities

None.

Return Value

The function result is a coded character set identifier (CCSID), or 0 if i5/OS PASE CCSID information is not available (such as when no i5/OS PASE program is running in the job).

Related Information

- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20
- "Qp2jobCCSID()—Retrieve Job CCSID for i5/OS PASE" on page 17

API introduced: V5R2

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Qp2ptrsize()—Retrieve i5/OS PASE Pointer Size

Syntax

```
#include <qp2user.h>

size_t Qp2ptrsize(void);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: Yes

Qp2ptrsize() returns the pointer size, in bytes, for the i5/OS Portable Application Solutions Environment (i5/OS PASE) program currently running in the job.

Parameters

None.

Authorities

None.

Return Value

The function result is 4 for a 32-bit program, or 8 for a 64-bit program. The result is zero if i5/OS PASE pointer size is not available (such as when no i5/OS PASE program is running in the job).

Related Information

- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20

API introduced: V5R2

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Qp2RunPase()—Run an i5/OS PASE Program

Syntax

```
#include <qp2user.h>

int Qp2RunPase(const char         *pathName,
               const char         *symbolName,
               const void         *symbolData,
               unsigned int       symbolDataLen,
               int                ccsid,
               const char  *const *argv,
               const char  *const *envp);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: No

The Qp2RunPase() function runs an i5/OS Portable Application Solutions Environment (i5/OS PASE) program in the job where the API is called. It loads the i5/OS PASE program and any necessary shared libraries and then transfers control to the program. Control returns to the caller when the i5/OS PASE program exits, terminates due to a signal, or returns without exiting.

Parameters

pathName
(Input) Pointer to a null-terminated character string that identifies the stream file in the Integrated File System that contains the i5/OS PASE program to run. The pathName string may include an absolute or relative path qualifier in addition to the stream file name. Relative path names are resolved using the current working directory.

symbolName
(Input) This argument must be a null pointer.

symbolData
(Input) This argument is ignored.

symbolDataLen
(Input) This argument is ignored.

ccsid (Input) The coded character set identifier (CCSID) initially used by the i5/OS PASE program. ccsid must specify a single-byte encoding (normally an ASCII CCSID) that i5/OS can convert to and from the job default CCSID, or a value of 1208 to indicate that the i5/OS PASE program uses UTF-8 encoding.

The system uses ccsid to set the CCSID of any bytestream file created by the i5/OS PASE program, and also to control character encoding conversions done for i5/OS PASE runtime interfaces that use i5/OS services.

argv (Input) Pointer to an array of pointers to null-terminated character strings that are passed as arguments to the i5/OS PASE program. The last element in the array must be a null pointer. An error is reported if the argv parameter pointer is null.

The system copies argument strings into i5/OS PASE memory and converts them from the job default CCSID to the CCSID specified by the ccsid parameter. By convention, the first argument string passed to an i5/OS PASE program should be the same as the pathName string.

envp (Input) Pointer to an array of pointers to null-terminated character strings that are passed as environment variables to the i5/OS PASE program. The last element in the array must be a null pointer. envp can be a null pointer if no environment variables need to be initialized for the i5/OS PASE program.

The system copies environment variable strings into i5/OS PASE memory and converts them from the job default CCSID to the CCSID specified by the ccsid parameter. By convension, environment variable strings take the form ″NAME=value″.

Authorities

Object Referred to	Authority Required
Each directory in the path to the i5/OS PASE program and shared libraries	*X
i5/OS PASE program (not a shell script) in a local file system	*X
i5/OS PASE program in a remote file system or shell script	*RX
i5/OS PASE shared library	*R

Return Value

The function result may be one of these special values:

QP2RUNPASE_ERROR (-1)	An internal error occurred during Qp2RunPase processing.
QP2RUNPASE_RETURN_NOEXIT (-2)	The i5/OS PASE program returned without exiting (by calling the i5/OS PASE **_RETURN** function).

If the result is not one of the special values above, it is a value that contains status information about how the i5/OS PASE program ended, in the same format as the stat_val parameter for the ILE waitpid function. You can use these macros in file <sys/wait.h> to interpret such a result:

WIFEXITED(stat_val)	Evaluates to a nonzero value if i5/OS PASE program ended normally.
WEXITSTATUS(stat_val)	If the value of the WIFEXITED(stat_val) is nonzero, evaluates to the low-order 8 bits of the value the i5/OS PASE program specified as the argument to exit or the function result returned by main.
WIFSIGNALED(stat_val)	Evaluates to a nonzero value if i5/OS PASE program ended because of the receipt of a terminating signal that was not caught by the process.
WTERMSIG(stat_val)	If the value of WIFSIGNALED(stat_val) is nonzero, evaluates to the number of the i5/OS PASE signal that caused the program to end. i5/OS PASE programs use the same signal numbers as AIX (which differ from ILE signal numbers).

Error Messages

Message ID	Error Message Text
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFB9C0 E	Error loading program &1. See previous messages.
CPFB9C1 E	System support for i5/OS Portable Application Solutions Environment not available.
CPFB9C2 E	Hardware support for i5/OS Portable Application Solutions Environment not available.
CPFB9C3 E	i5/OS PASE CCSID and job default CCSID are incompatible.
CPFB9C7 E	i5/OS PASE already running in this job.
CPFB9C8 E	File descriptors 0, 1, and 2 must be open to run the i5/OS PASE program.
CPFB9CB E	Qp2RunPase second argument must be a null pointer.

Usage Notes

1. Qp2RunPase works like the AIX execve function, including the ability to run shell scripts and the rules for resolving shared libraries (which may include using i5/OS PASE environment variable LIBPATH).
2. If an absolute path (starting with ″/″) is specified for the pathName string or in the first line of a shell script identified by pathName and that path cannot be opened or is not a regular bytestream file, the system generally searches the /QOpenSys file system for the file. See environment variable **PASE_EXEC_QOPENSYS** in “i5/OS PASE Environment Variables” on page 79 for more information.
3. Qp2RunPase cannot run an i5/OS PASE program or shared library stored in a file system that is not threadsafe in a job that is multithread capable. Any job started by the i5/OS PASE fork function is multi-thread capable.
4. You can set these ILE environment variables before calling Qp2RunPase to control the i5/OS PASE operation:

QIBM_USE_DESCRIPTOR_STDIO	When this ILE environment variable is set to Y or I, both i5/OS PASE runtime and ILE C runtime use Integrated File System file descriptors 0, 1, and 2 for stdin, stdout, and stderr. Otherwise, i5/OS PASE file descriptors 0, 1, and 2 are mapped to ILE C runtime files stdin, stdout, and stderr (which may not use any Integrated File System file descriptors). i5/OS PASE and ILE generally use different descriptor numbers for the same open file, but when QIBM_USE_DESCRIPTOR_STDIO is set to Y or I, any operation against i5/OS PASE file descriptors 0, 1, or 2 is also done for the same Integrated File System file descriptor number so i5/OS PASE and ILE C use the same files for stdin, stdout, and stderr.
QIBM_PASE_DESCRIPTOR_STDIO	This ILE environment variable controls ASCII/EBCDIC conversion for data read or written through i5/OS PASE files stdin, stdout, and stderr to Integrated File System file descriptors 0, 1, and 2. ASCII/EBCDIC conversion is always done (and this variable is ignored) unless QIBM_USE_DESCRIPTOR_STDIO is set to either Y or I. If QIBM_PASE_DESCRIPTOR_STDIO is set to B, the PASE program processes binary data (without ASCII/EBCDIC conversion). Otherwise, ASCII/EBCDIC conversion is done for any data read from or written to i5/OS PASE file descriptors 0, 1, or 2.
QIBM_PASE_FLUSH_STDIO	This ILE environment variable controls whether i5/OS PASE runtime flushes every write to a standard output stream attached to a Data Management file (such as a spooled printer file) or to the Dynamic Screen Manager in an interactive job. QIBM_PASE_FLUSH_STDIO must be set before starting i5/OS PASE, and only applies when i5/OS PASE is NOT using IFS descriptors for standard I/O (QIBM_USE_DESCRIPTOR_STDIO is not set). It is usually only needed for interactive programs that require immediate display of output that does not end with newline. These values are supported: **Y** flush both stdout and stderr **1** flush only stdout (i5/OS PASE descriptor 1) **2** flush only stderr (i5/OS PASE descriptor 2)
QIBM_PASE_USE_PRESTART_JOBS	When this ILE environment variable is set to Y, i5/OS PASE runtime uses prestarted jobs for child processes created by fork and for any job started by the systemCL i5/OS PASE runtime function (to run a CL command). You should add prestarted job entries (ADDPJE command) for programs QP0ZSPWT (used by fork) and QP0ZSPWP (used by systemCL) to any subsystem description that will run jobs that use this support.

5. i5/OS PASE environment variables are independent of ILE environment variables. See “i5/OS PASE Environment Variables” on page 79 for more information, including i5/OS PASE environment variables you can set to control runtime behaviors that differ from AIX.
6. The ccsid parameter provides the initial i5/OS PASE CCSID value, but the i5/OS PASE program can use the _SETCCSID function to change the i5/OS PASE CCSID or to rebind to a change in the job default CCSID. The i5/OS PASE CCSID should generally be the CCSID equivalent of the code set for the current locale. See “i5/OS PASE Locales” on page 70 to determine what locales are supported by i5/OS PASE.
7. You may want to increase the number of file descriptors in the job by calling DosSetRelMaxFH before you call Qp2RunPase. By default, i5/OS jobs support only 200 open file descriptors, while i5/OS PASE programs generally expect to be able to open 32 767 file descriptors, and the system requires file descriptors to open bytestream files that contain the i5/OS PASE program and any shared libraries it uses.
8. You may want to establish Qp2SignalPase as the handler for any ILE signal that needs to be visible to the i5/OS PASE program. For example, system support for Sockets (used by i5/OS PASE runtime) only sends SIGIO and SIGURG as ILE signals, so ILE signal handling must be set up before calling an i5/OS PASE program that relies on SIGIO or SIGURG as i5/OS PASE signals. i5/OS PASE runtime automatically establishes Qp2SignalPase as the handler for every ILE signal in a fork child process.
9. You may want to call ILE interfaces pthread_setcancelstate and pthread_setcanceltype to set pthread cancel state and cancel type before calling Qp2RunPase in a process that did prior pthread work. i5/OS PASE pthreads use ILE pthreads and Qp2RunPase assumes that ILE pthread cancel state and cancel type are set to defaults (PTHREAD_CANCEL_ENABLE and PTHREAD_CANCEL_DEFERRED). The state of these attributes when a program ends is whatever value was last set by either ILE or i5/OS PASE code.
10. Time-of-day information in an i5/OS PASE program depends on the value of i5/OS PASE environment variable TZ, which provides information about timezone name and offset from UTC (Universal Coorodinated Time). For example, the correct TZ setting for Central Time in the USA is TZ=CST6CDT. See AIX documentation for more information about environment varble TZ.
11. Any credentials changes (user, group, or group list changes) made by an i5/OS PASE program are generally persistent in the job. The job (thread) credentials before and after a call to Qp2RunPase may not be the same if the i5/OS PASE program calls any of the setuid or setgid family of interfaces. However, the system saves credentials before running any i5/OS PASE program with the S_ISUID or S_ISGID attribute, and automatically restores the saved credentials before returning to the caller of Qp2RunPase.
12. Character conversions controlled by the ccsid parameter only handle the single-byte component of an EBCDIC-mixed CCSID (for the job default CCSID). This restricts the i5/OS PASE program name specified by the pathName parameter, argument strings passed through the argv parameter, and environment variables passed through the envp parameter to single-byte characters.
13. If an i5/OS PASE program needs to use DBCS characters for i5/OS PASE runtime functions such as file system interfaces, it must run with the i5/OS PASE CCSID (ccsid parameter) set to 1208 because i5/OS PASE runtime provides complete support for DBCS characters using UTF-8 encoding only.
14. Older versions of **Qp2RunPase** used *symbolName*, *symbolData*, and *symbolDataLen* to pass inputs other than character string arguments and environment variables to the i5/OS PASE program. An i5/OS PASE program can retrieve any inputs that cannot be expressed as null-terminated strings (such as tagged pointers) by calling ILE or OPM code (using _ILECALL or _PGMCALL) with by-address arguments.

Related Information

- The <sys/wait.h> file (see Header Files for UNIX-Type Functions)
- DosSetRelMaxFH()—Change Maximum Number of File Descriptors
- pthread_setcancelstate()—Set Cancel State

- pthread_setcanceltype()—Set Cancel Type
- "QP2SHELL() and QP2SHELL2()—Run an i5/OS PASE Shell Program" on page 2
- "QP2TERM()—Run an i5/OS PASE Terminal Session" on page 5

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Qp2SignalPase()—Post an i5/OS PASE Signal

Syntax

```
#include <qp2user.h>

int Qp2SignalPase(int   signo);
```

Service Program Name: QP2USER
Default Public Authority: *USE
Threadsafe: Yes

The Qp2SignalPase() function posts an i5/OS Portable Application Solutions Environment (i5/OS PASE) signal to an i5/OS PASE program that is already running in the job.

Parameters

signo (Input) Signal number to post. A positive value is an ILE signal number, which causes the system to post a corresponding i5/OS PASE signal. ILE and i5/OS PASE signals correspond if they have the same name (for example, SIGTERM) in a system-provided header file. A negative value is the negation of an i5/OS PASE (and AIX) signal number.

Authorities

None.

Return Value

The function result is an integer that indicates whether the i5/OS PASE signal was posted successfully. Header file qp2user.h defines the following constants for the return code from Qp2SignalPase:

QP2CALLPASE_NORMAL(0)	An i5/OS PASE signal was posted successfully.
QP2CALLPASE_ENVIRON_ERROR(2)	The operation is not allowed because no i5/OS PASE program is running in the job, or the thread that called Qp2CallPase was neither the initial i5/OS PASE thread nor a thread created using i5/OS PASE pthread interfaces.
QP2CALLPASE_ARG_ERROR(4)	The signo parameter value is invalid.
QP2CALLPASE_TERMINATING(6)	The i5/OS PASE program is terminating. No function result was returned. The i5/OS PASE program may have run the exit function, or a signal might have caused the program to terminate.

Usage Notes

1. Qp2SignalPase is supported only when an i5/OS PASE program is currently running in the job. This means that Qp2RunPase must be actively called in the job, or the job must be a fork child process.
2. Not all ILE signals have an i5/OS PASE equivalent and Qp2SignalPase never converts ILE SIGCHLD to a corresponding PASE signal. This special handling for SIGCHLD avoids duplicate PASE signals for the termination of a single child process (because the system may send both ILE and i5/OS PASE signals to the parent of any fork child process that ends).

3. If there is only one i5/OS PASE thread running in the job, the signal remains pending until control is transferred to the i5/OS PASE program. If other i5/OS PASE threads are running at the time Qp2SignalPase is called, the system may chose one of the other threads to deliver the signal.

Related Information

- "Qp2CallPase()—Call an i5/OS PASE Procedure" on page 8
- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Runtime Functions For Use by i5/OS PASE Programs

i5/OS(TM) PASE runtime includes interfaces supported on AIX(R) and interfaces unique to i5/OS PASE. They are unique to i5/OS PASE

The runtime functions are:

- "_CVTERRNO()—Convert ILE errno to i5/OS PASE errno" on page 44 (Convert ILE errno to i5/OS PASE errno) converts an ILE errno value to a corresponding i5/OS PASE errno value.
- "_CVTSPP()—Convert Space Pointer for i5/OS PASE" on page 45 (Convert Space Pointer for i5/OS PASE) converts a tagged space pointer value to an equivalent i5/OS PASE memory address.
- "_CVTTS64()—Convert Teraspace Address for i5/OS PASE" on page 46 (Convert Teraspace Address for i5/OS PASE) converts a 64-bit teraspace address to an equivalent i5/OS PASE memory address.
- "_GETTS64() and _GETTS64_SPP()—Get Teraspace Address for i5/OS PASE" on page 47 (Get Teraspace Address for i5/OS PASE) returns the 64-bit teraspace address equivalent of an i5/OS PASE memory address.
- "_GETTS64() and _GETTS64_SPP()—Get Teraspace Address for i5/OS PASE" on page 47 (Get Teraspace Address (from Space Pointer) for i5/OS PASE) returns the 64-bit teraspace address stored in a 16-byte space pointer.
- "_GETTS64M()—Get Multiple Teraspace Pointers for i5/OS PASE" on page 48 (Get Multiple Teraspace Pointers for i5/OS PASE) retrieves teraspace address equivalents for a set of i5/OS PASE memory addresses.
- "_ILECALLX()—Call an ILE Procedure for i5/OS PASE" on page 49 (Call an ILE Procedure for i5/OS PASE) allows an i5/OS PASE program to call an ILE procedure.
- » "_ILEKILL()—Send ILE Signal for i5/OS PASE" on page 30 (Send ILE Signal for i5/OS PASE) invokes the ILE kill function to send an ILE signal to a process or process group. «
- "_ILELOADX()—Load an ILE Bound Program for i5/OS PASE" on page 53 (Load an ILE Bound Program for i5/OS PASE) allows an i5/OS PASE program to load (activate) an ILE bound program.
- "_ILELOADX()—Load an ILE Bound Program for i5/OS PASE" on page 53 (Load an ILE Bound Program for i5/OS PASE) loads (activates) an ILE-bound program.
- "_ILESYMX()—Find an Exported ILE Symbol for i5/OS PASE" on page 54 (Find Exported ILE Symbol for i5/OS PASE) allows an i5/OS PASE program to get a tagged pointer to the data or procedure exported for a symbol exported by an ILE activation.
- "_ILESYMX()—Find an Exported ILE Symbol for i5/OS PASE" on page 54 (Find an Exported ILE Symbol for i5/OS PASE) finds an exported ILE symbol in the activation of an ILE-bound program.
- "_MEMCPY_WT()—Copy Memory With Tags for i5/OS PASE" on page 56 (Copy Memory With Tags for i5/OS PASE) allows an i5/OS PASE program to copy memory with tagged pointers.
- » "_OPEN_CCSID()—Open With CCSID for i5/OS PASE" on page 31 (Open With CCSID for i5/OS PASE) opens a file with CCSID tagging. «

- "_PGMCALL()—Call an i5/OS Program for i5/OS PASE" on page 57 (Call an i5/OS Program for i5/OS PASE) calls an i5/OS program (object type *PGM) from an i5/OS PASE program.
- "_RETURN()—Return Without Exiting i5/OS PASE" on page 59 (Return without Exiting i5/OS PASE) returns to the ILE called that called i5/OS PASE in this job, without exiting the i5/OS PASE program.
- "_RSLOBJ()—Resolve to an i5/OS Object for i5/OS PASE" on page 60 (Resolve to an i5/OS Object for i5/OS PASE) resolves to an i5/OS object.
- "_SETCCSID()—Set i5/OS PASE CCSID" on page 62 (Set i5/OS PASE CCSID) retrieves and sets the i5/OS PASE Coded Character Set Identifier (CCSID) value.
- "_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE" on page 64 (Set Space Pointer for i5/OS PASE) sets a tagged space pointer to the teraspace equivalent of an i5/OS PASE memory address.
- "_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE" on page 64 (Set Space Pointer for i5/OS PASE) sets a space pointer from teraspace address for i5/OS PASE.
- "_SETSPPM()—Set Multiple Space Pointers for i5/OS PASE" on page 65 (Set Multiple Space Pointers for i5/OS PASE) Sets multiple space pointers for i5/OS PASE.
- "_STRLEN_SPP()—Determine Character String Length for i5/OS PASE" on page 66 (Determine Character String Length for i5/OS PASE) determines the length of a null-terminated character string.
- "_STRNCPY_SPP()—Copy Character String for i5/OS PASE" on page 67 (Copy Character String for i5/OS PASE) copies a null-terminated character string.
- "build_ILEarglist()—Build an ILE Argument List for i5/OS PASE" on page 27 (Build an ILE Argument List for i5/OS PASE) builds an ILE argument list using argument values copied from an i5/OS PASE function with the same signature.
- "fork400()and f_fork400()—Create A New Process with i5/OS PASE Options" on page 28 (Create a New Process with i5/OS PASE Options) creates a new (child) process that is an almost exact copy of the calling (parent) process.
- "fork400()and f_fork400()—Create A New Process with i5/OS PASE Options" on page 28 (Create a New Process with i5/OS PASE Options) creates a new (child) process that is an almost exact copy of the calling (parent) process.
- "QMHRCVM()—Receive Nonprogram Message for i5/OS PASE" on page 32 (Receive Nonprogram Message for i5/OS PASE) allows an i5/OS PASE program to receive a message from a nonprogram message queue.
- "QMHRCVPM()—Receive Program Message for i5/OS PASE" on page 33 (Receive Program Message for i5/OS PASE) allows an i5/OS PASE program to receive a message from a program call message queue or from the job external message queue.
- "QMHSNDM()—Send Nonprogram Message for i5/OS PASE" on page 35 (Send Nonprogram Message for i5/OS PASE) allows an i5/OS PASE program to send a message to a nonprogram message queue so it can communicate with another job or user.
- "QMHSNDPM()—Send Program Message for i5/OS PASE" on page 37 (Send Program Message for i5/OS PASE) allows an i5/OS PASE program to send a message to a program call message queue or to the job external message queue.
- "Qp2jobCCSID()—Retrieve Job CCSID for i5/OS PASE" on page 17 (Retrieve Job CCSID for i5/OS PASE) returns the job default CCSID from the last time the i5/OS PASE CCSID was set.
- "Qp2paseCCSID()—Retrieve i5/OS PASE CCSID" on page 18 (Retrieve i5/OS PASE CCSID) returns the i5/OS PASE CCSID from the last time the i5/OS PASE CCSID was set.
- "Qp2setenv_ile()—Set ILE environment variables for i5/OS PASE" on page 38 (Set ILE Environment Variables for i5/OS PASE) allows an i5/OS PASE program to set ILE environment variables.
- "size_ILEarglist()—Compute ILE Argument List Size for i5/OS PASE" on page 40 (Compute ILE Argument List Size for i5/OS PASE) computes the number of bytes of memory required to build an ILE argument list.

- “SQLOverrideCCSID400()—Override SQL CLI CCSID for i5/OS PASE” on page 41 (Override SQL CLI CCSID for i5/OS PASE) allows an i5/OS PASE program to specify a CCSID for character arguments and results on SQL runtime functions.
- “systemCL()—Run a CL Command for i5/OS PASE” on page 42 (Run a CL Command for i5/OS PASE) allows an i5/OS PASE program to run a CL command.

Top | “i5/OS PASE APIs,” on page 1 | APIs by category

build_ILEarglist()—Build an ILE Argument List for i5/OS PASE

Syntax

```
#include <as400_protos.h>

int build_ILEarglist(ILEarglist_base   *ILEarglist,
                     const void        *PASEarglist,
                     const arg_type_t  *signature);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **build_ILEarglist()** function builds an ILE argument list using argument values copied from an i5/OS PASE function with the same signature.

Parameters

ILEarglist
(Output) Pointer to a 16-byte aligned buffer allocated by the caller for the ILE argument list. ILEarglist must be long enough to contain all arguments specified in the signature list.

PASEarglist
(Input) Pointer to the first argument passed to an i5/OS PASE function that accepts arguments equivalent to those specified by the signature list.

signature
(Input) Pointer to a list of arg_type_t values that specify the sequence and type of arguments passed to the ILE procedure. ILE procedures can accept a maximum of 400 arguments. The actual number of arguments processed by the build_ILEarglist function is determined by the number of entries in the signature list, which is determined by the location of the first ARG_END value in the list. The following values are supported in the signature list:

ARG_END(0)	Specifies the end of the signature list.
ARG_INT8 (-1)	Signed 1-byte integer argument.
ARG_UINT8 (-2)	Unsigned 1-byte integer argument.
ARG_INT16 (-3)	Signed 2-byte integer argument.
ARG_UINT16 (-4)	Unsigned 2-byte integer argument.
ARG_INT32 (-5)	Signed 4-byte integer argument.
ARG_UINT32 (-6)	Unsigned 4-byte integer argument.
ARG_INT64 (-7)	Signed 8-byte integer argument.
ARG_UINT64 (-8)	Unsigned 8-byte integer argument.
ARG_FLOAT32 (-9)	4-byte floating-point argument.
ARG_FLOAT64 (-10)	8-byte floating-point argument.

ARG_MEMPTR (-11)	The argument is a memory address. The i5/OS PASE procedure argument is an i5/OS PASE memory address that build_ILEarglist copies into the ILEpointer type value in the ILE argument list. See "_ILECALLX()—Call an ILE Procedure for i5/OS PASE" on page 49 (_ILECALLX) for more information about how ARG_MEMPTR arguments are handled.
ARG_MEMTS64 (-14)	The argument is a memory address. The i5/OS PASE procedure argument is an i5/OS PASE memory address that build_ILEarglist copies into the ts64_t type value in the ILE argument list. See "_ILECALLX()—Call an ILE Procedure for i5/OS PASE" on page 49 (_ILECALLX) for more information about how ARG_MEMTS64 arguments are handled.
ARG_TS64PTR (-15)	The argument is a 64-bit teraspace pointer.
Any positive number (1-32767)	The argument is an aggregate (structure or union). The value in the signature list is the length, in bytes, of the aggregate.

Authorities

build_ILEarglist requires no authority.

Return Value

build_ILEarglist returns the number of bytes used to build the ILE argument list (including storage for the ILEarglist_base type), or zero if an error was detected in the input arguments.

Usage Notes

1. **build_ILEarglist** does no character encoding conversions, so the i5/OS PASE program may need to convert argument and result character strings between ASCII and EBCDIC. i5/OS PASE runtime function **iconv** can be used for character conversions.
2. **build_ILEarglist** does not support argument types ARG_SPCPTR or ARG_OPENPTR (which are supported by **_ILECALLX**) because the AIX Application Binary Interface for PowerPC provides no way to ensure 16-byte alignment for arguments pushed onto the stack.
3. **build_ILEarglist** does not directly support aggregate function results. You need to set result.r_aggregate.addr in the PASEarglist structure to the address of a buffer where the ILE procedure will store the aggregate result.
4. Older versions of **build_ILEarglist** accepted additional arguments in an attempt to handle aggregate function results, but those arguments were removed because they cannot be supported reliably. If you need to compile source that passes the additional arguments, you must define macro **OLD_build_ILEarglist** and include <as400_types.h> to access the old support.

Related Information

- "_ILECALLX()—Call an ILE Procedure for i5/OS PASE" on page 49
- "size_ILEarglist()—Compute ILE Argument List Size for i5/OS PASE" on page 40

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

fork400()and f_fork400()—Create A New Process with i5/OS PASE Options

Syntax

```
#include <as400_protos.h>

pid_t fork400(const char   *jobname,
              unsigned int  resourceID);

pid_t f_fork400(const char   *jobname,
                unsigned int  resourceID);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **fork400()** function creates a new process. The new process (the child process) is an almost exact copy of the calling process (the parent process). **fork400()** is called once (by the parent process), but returns twice (once in the parent process and once in the child process). **fork400()** is the same as the **fork()** function plus it allows additional i5/OS PASE unique options to be specified.

f_fork400() function is a similarly enhanced version of the **f_fork()** function. When **f_fork400()** (or **f_fork()**) is used, one of the **exec** functions must be called in the child process immediately after it is created. **f_fork400()** does not call the fork handlers so the application data, mutexes and the locks are all undefined in the child process.

Parameters

jobname
: (Input) Pointer to a null-terminated string in the i5/OS PASE CCSID that specifies the i5/OS job name of the new process.

  The job name specified must begin with an alphabetic character [A-Z] or the characters [$#@]. The remaining characters must be alphanumeric [A-Z] or [0-9] or [$#@_.]. The string should not be longer than 10 characters (not including the terminating null character). If the specified jobname is invalid, the jobname parameter value is ignored.

resourceID
: (Input) A positive integer value specifying the resources affinity identifier for the new process.

  Use the value of 0 to let the operating system select the resources affinity identifier value automatically.

Authorities

fork400() and **f_fork400()** require no authority.

Return Value

Upon successful completion, the **fork400()** or **f_fork400()** function returns a value of 0 to the child process and the process ID of the child process to the parent process. Otherwise, a value of -1 is returned to the parent process, no child process is created, and the errno global variable is set to indicate the error.

Error Conditions

At least these errno values can be returned, with other values also possible (such as i5/OS-unique ILE errno EAPAR):

[EAGAIN]	Exceeds the limit on the total number of processes running or the system does not have the resources necessary to create another process.
[ENOMEM]	Not enough space exists for this process.

[EINVAL]	An invalid argument value was specified.

Usage Notes

1. Consult the AIX documentation for **fork()** and **f_fork()** for additional details regarding attributes of the parent process inherited by the child process and differences between **fork()** and **f_fork()**.
2. The i5/OS PASE environment specification *QIBM_PASE_USE_PRESTART_JOBS=Y* will be ignored when the fork400() or f_fork400() functions are used with a non-null jobname or a non-zero resourceID value.

Related Information

- See the "i5/OS PASE Environment Variables" on page 79 documentation for information about the *PASE_FORK_JOBNAME* environment variable that can be used to specify the i5/OS job name for new processes created using the **fork()** or **f_fork()** functions.

API introduced: V5R3

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_ILEKILL()—Send ILE Signal for i5/OS PASE

Syntax

```
#include <as400_protos.h>

int _ILEKILL(pid_t pid,
             int   signo);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: No

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **_ILEKILL** invokes the ILE **kill** function to send an ILE signal to a process or process group.

Parameters

pid (Input) Specifies the identifier of a process or process group. See documentation for the ILE kill()—Send Signal to Process or Group of Processes function for more information.

signo (Input) Specifies the signal to send. A *signo* value greater than zero is interpreted as an AIX signal number that the system converts to an equalent ILE signal number. For example, the AIX signal number for SIGTERM (15) is converted to the ILE signal number for SIGTERM (6). AIX signals that have no ILE equivalent return an error.

A *signo* value less than zero is negated to determine the ILE signal number, and *signo* zero simply checks that the target process or process group exists.

Authorities

See documentation for the ILE kill()—Send Signal to Process or Group of Processes function for information about required authorities.

Return Value

_ILEKILL returns zero for normal completion, or -1 wih an **errno** for any error.

Error Conditions

See ILE kill()—Send Signal to Process or Group of Processes for more information about error conditions.

Related Information

- kill()—Send Signal to Process or Group of Processes

«

API introduced: V5R4

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_OPEN_CCSID()—Open With CCSID for i5/OS PASE

Syntax

```
#include <as400_protos.h>

int _OPEN_CCSID(const char *path,
                int        oflags,
                mode_t     mode,
                int        ccsid);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: No

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **_OPEN_CCSID** function establishes a connection between the file named by the *path* parameter and a file descriptor. **_OPEN_CCSID** works the same as the i5/OS PASE **open** function, except that any file it creates is tagged with a specified Coded Character Set Identifier (CCSID).

Parameters

path (Input) Specifies the address of a null-terminated character string (in the current i5/OS PASE CCSID) that contains the path name of the file to open.

oflags (Input) Specifies the the type of access, special open processing, type of update, and initial state of the open file. See AIX documentation for the **open** function for more information about **oflags**.

mode (Input) the read, write, and execute permissions of the file to be created (requested by the O_CREAT flag). If the file already exists, this parameter is ignored. See AIX documentation for the **open** function for more information about **mode**.

ccsid (Input) Specifies the i5/OS PASE CCSID value that describes the data that will be stored in a file created by **_OPEN_CCSID**. If zero is specified, the current i5/OS PASE CCSID is used. If the file already exists, this parameter is ignored.

Authorities

See documenation for the ILE Open File (open) function for information about required authorities.

Return Value

_OPEN_CCSID returns either the file descriptor for the open file, or -1 if an error occurred. The **errno** variable is set to indentify the specific error.

Error Conditions

See AIX documentation for the **open** function and ILE Open File (open) function for more information about error conditions.

Usage Notes

1. ILE runtime uses the CCSID tag for a file to control automatic conversion for data read or written through a file opened in text mode. Any data an i5/OS PASE program writes to or reads from a file descriptor is generally *not* converted. The only exception is for the initial file descriptors 0, 1, and 2 provided when the Qp2RunPase API is called to start an i5/OS PASE program, which default to converting file data between the i5/OS PASE CCSID and the job default CCSID (see "Qp2RunPase()—Run an i5/OS PASE Program" on page 20 (Qp2RunPase) for more information).
2. Other than special support for file descriptors 0, 1, and 2, i5/OS PASE runtime does no CCSID conversion of file data. This differs from ILE runtime, which does CCSID conversion between the file CCSID and job default CCSID for any file opened in text mode.
3. The i5/OS PASE runtime functions cstoccsid and ccsidtocs convert between AIX Character Set names and CCSID values.

Related Information

- "_SETCCSID()—Set i5/OS PASE CCSID" on page 62—Set i5/OS PASE CCSID

API introduced: V5R4

Top | "i5/OS PASE APIs," on page 1 | APIs by category

QMHRCVM()—Receive Nonprogram Message for i5/OS PASE

Syntax

```
#include <os400msg.h>

int QMHRCVM(void         *msginfo,
            int          msginfoLen,
            const char   *format,
            const void   *msgq,
            const char   *msgtype,
            int          *msgkey,
            int          wait,
            const char   *action,
            void         *errcode);

int QMHRCVM1(void        *msginfo,
             int         msginfoLen,
             const char  *format,
             const void  *msgq,
             const char  *msgtype,
             int         *msgkey,
             int         wait,
             const char  *action,
             void        *errcode,
             int         ccsid);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The Receive Nonprogram Message (QMHRCVM and QMHRCVM1) i5/OS PASE runtime functions allow an i5/OS PASE program to receive a message from a nonprogram message queue.

Parameters

These i5/OS PASE runtime functions accept the same arguments as the Receive Nonprogram Message (QMHRCVM) i5/OS API, except that the i5/OS PASE functions use character string inputs that are null-terminated strings in the i5/OS PASE CCSID. i5/OS PASE runtime automatically converts input character strings to the job default CCSID and pads with blanks (as necessary) to match the fixed-length inputs required by the system API.

No conversions are done by i5/OS PASE runtime for the msginfo and errcode (input/output) arguments because they can contain a mixture of character and binary data. The ccsid argument specifies the CCSID for character data returned by the system API in the msginfo argument, and users can request CCSID information for the errcode argument by using ERRC0200 format. The QMHRCVM i5/OS PASE runtime function uses a default for the ccsid value passed to the system API that does not do any CCSID conversion for character data in the received message.

See QMHRCVM()—Receive Nonprogram Message for further description of the arguments for the **QMHRCVM** and **QMHRCVM1** i5/OS PASE runtime functions.

Authorities

See QMHRCVM()—Receive Nonprogram Message for information about authorities required for the **QMHRCVM** and **QMHRCVM1** i5/OS PASE runtime functions.

Return Value

The function result is zero for normal completion. The result is nonzero if any input character string could not be converted to the job default CCSID or was too long for the QMHRCVM API, or if the QMHRCVM API returned error information in the errcode argument.

Related Information

- QMHRCVM()—Receive Nonprogram Message (system API)
- "QMHRCVM()—Receive Nonprogram Message for i5/OS PASE" on page 32
- "QMHSNDPM()—Send Program Message for i5/OS PASE" on page 37
- "QMHRCVPM()—Receive Program Message for i5/OS PASE"

API introduced: V5R1

Top | "i5/OS PASE APIs," on page 1 | APIs by category

QMHRCVPM()—Receive Program Message for i5/OS PASE

Syntax

```
#include <os400msg.h>

int QMHRCVPM(void         *msginfo,
             int          msginfoLen,
             const char   *format,
             const char   *pgmq,
             int          pgmqDelta,
             const char   *msgtype,
             int          *msgkey,
             int          wait,
             const char   *action,
             void         *errcode);

int QMHRCVPM1(void        *msginfo,
              int         msginfoLen,
              const char  *format,
              const char  *pgmq,
              int         pgmqDelta,
              const char  *msgtype,
              int         *msgkey,
              int         wait,
              const char  *action,
              void        *errcode,
              int         pgmqLen,
              const char  *pgmqQual);

int QMHRCVPM2(void        *msginfo,
              int         msginfoLen,
              const char  *format,
              const void  *pgmq,
              int         pgmqDelta,
              const char  *msgtype,
              int         *msgkey,
              int         wait,
              const char  *action,
              void        *errcode,
              int         pgmqLen,
              const char  *pgmqQual,
              const char  *pgmqType,
              int         ccsid);
```

Public Default Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The Receive Program Message (QMHRCVPM, QMHRCVPM1, and QMHRCVPM2) i5/OS PASE runtime functions allow an i5/OS PASE program to receive a message from a program call message queue or from the job external message queue.

Parameters

These i5/OS PASE runtime functions accept the same arguments as the Receive Program Message (QMHRCVPM) i5/OS API, except that the i5/OS PASE functions use character string inputs that are null-terminated strings in the i5/OS PASE CCSID. i5/OS PASE runtime automatically converts input character strings to the job default CCSID and pads with blanks (as necessary) to match the fixed-length inputs required by the system API.

No conversions are done by i5/OS PASE runtime for the msginfo and errcode (input/output) arguments because they can contain a mixture of character and binary data. The ccsid argument specifies the CCSID for character data returned by the system API in the msginfo argument, and users can request CCSID information for the errcode argument by using ERRC0200 format. The QMHRCVPM and QMHRCVPM1 i5/OS PASE runtime functions use a default for the ccsid value passed to the system API that does not do any CCSID conversion for character data in the received message.

See QMHRCVPM()—Receive Program Message for further description of the arguments for the QMHRCVPM, QMHRCVPM1, and QMHRCVPM2 i5/OS PASE runtime functions.

Authorities

See QMHRCVPM()—Receive Program Message for information about authorities required for the QMHRCVPM, QMHRCVPM1, and QMHRCVPM2 i5/OS PASE runtime functions.

Return Value

The function result is zero for normal completion. The result is nonzero if any input character string could not be converted to the job default CCSID or was too long for the QMHRCVPM API, or if the QMHRCVPM API returned error information in the errcode argument.

Usage Notes

1. The system only creates program call message queues ILE procedures and OMI programs, so you cannot send to or receive from a program message queue for a specific function in an i5/OS PASE program.
2. When "*" is specified for the pgmq argument, the system locates the program call message queue for an (internal) ILE procedure in service program QP2USER that is the apparent caller of any ILE procedure called by the i5/OS PASE program using i5/OS PASE runtime function _ILECALLX or _ILECALL. This queue is the target for messages a called ILE procedure sends to its caller, and is also used for machine exceptions caused by operations inside the i5/OS PASE program (such as message MCH0601 a for storage reference error).

Related Information

- QMHRCVPM()—Receive Program Message (system API)
- "QMHRCVPM()—Receive Program Message for i5/OS PASE" on page 33
- "QMHSNDM()—Send Nonprogram Message for i5/OS PASE"
- "QMHRCVM()—Receive Nonprogram Message for i5/OS PASE" on page 32

API introduced: V5R1

Top | "i5/OS PASE APIs," on page 1 | APIs by category

QMHSNDM()—Send Nonprogram Message for i5/OS PASE

Syntax

```
#include <os400msg.h>

int QMHSNDM(const char  *msgid,
            const char   *msgf,
            const void   *msgdata,
            int          msgdataLen,
            const char   *msgtype,
            const char   *msgqList,
            int          msgqCount,
            const char   *rpyq,
            int          *msgkey,
            void         *errcode);

int QMHSNDM1(const char  *msgid,
             const char  *msgf,
             const void  *msgdata,
             int         msgdataLen,
             const char  *msgtype,
             const char  *msgqList,
           int        msgqCount,
           const char *rpyq,
           int        *msgkey,
           void       *errcode,
           int        ccsid);
```

Public Default Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The Send Nonprogram Message (QMHSNDM and QMHSNDM1) i5/OS PASE runtime functions allow an i5/OS PASE program to send a message to a nonprogram message queue so it can communicate with another job or user.

Parameters

These i5/OS PASE runtime functions accept the same arguments as the Send Nonprogram Message (QMHSNDM) i5/OS API, except that the i5/OS PASE functions use character string inputs that are null-terminated strings in the i5/OS PASE CCSID. i5/OS PASE runtime automatically converts input character strings to the job default CCSID and pads with blanks (as necessary) to match the fixed-length inputs required by the system API.

No conversions are done for the msgdata (input) argument and the errcode (input/output) argument because they can contain a mixture of character and binary data. The *ccsid* argument specifies the CCSID for character data in the *msgdata* argument, and users can request CCSID information for the *errcode* argument by using ERRC0200 format. The QMHSNDM i5/OS PASE runtime function uses the current i5/OS PASE CCSID as a default for the *ccsid* value passed to the system API.

See QMHSNDM()—Send Nonprogram Message for further description of the arguments for the QMHSNDM and QMHSNDM1 i5/OS PASE runtime functions.

Authorities

See QMHSNDM()—Send Nonprogram Message for information about authorities required for the QMHSNDM and QMHSNDM1 i5/OS PASE runtime functions.

Return Value

The function result is zero for normal completion. The result is nonzero if any input character string could not be converted to the job default CCSID or was too long for the QMHSNDM API, or if the QMHSNDM API returned error information in the *errcode* argument.

Related Information

- QMHSNDM()—Send Nonprogram Message (system API)
- “QMHRCVM()—Receive Nonprogram Message for i5/OS PASE” on page 32
- “QMHSNDPM()—Send Program Message for i5/OS PASE” on page 37
- “QMHRCVPM()—Receive Program Message for i5/OS PASE” on page 33

API introduced: V5R1

Top | “i5/OS PASE APIs,” on page 1 | APIs by category

QMHSNDPM()—Send Program Message for i5/OS PASE

Syntax

```
#include <os400msg.h>

int QMHSNDPM(const char   *msgid,
             const char   *msgf,
             const void   *msgdata,
             int          msgdataLen,
             const char   *msgtype,
             const char   *pgmq,
             int          pgmqDelta,
             int          *msgkey,
             void         *errcode);

int QMHSNDPM1(const char  *msgid,
              const char  *msgf,
              const void  *msgdata,
              int         msgdataLen,
              const char  *msgtype,
              const char  *pgmq,
              int         pgmqDelta,
              int         *msgkey,
              void        *errcode,
              int         pgmqLen,
              const char  *pgmqQual,
              int         extWait);

int QMHSNDPM2(const char  *msgid,
              const char  *msgf,
              const void  *msgdata,
              int         msgdataLen,
              const char  *msgtype,
              const void  *pgmq,
              int         pgmqDelta,
              int         *msgkey,
              void        *errcode,
              int         pgmqLen,
              const char  *pgmqQual,
              int         extWait,
              const char  *pgmqType,
              int         ccsid);
```

Library: Standard C Library (libc.a)
Default Public Authority: *USE
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The Send Program Message (QMHSNDPM, QMHSNDPM1, and QMHSNDPM2) i5/OS PASE runtime functions allow an i5/OS PASE program to send a message to a program call message queue or to the job external message queue.

Parameters

These i5/OS PASE runtime functions accept the same arguments as the Send Program Message (QMHSNDPM) i5/OS API, except that the i5/OS PASE functions use character string inputs that are null-terminated strings in the i5/OS PASE CCSID. i5/OS PASE runtime automatically converts input character strings to the job default CCSID and pads with blanks (as necessary) to match the fixed-length inputs required by the system API.

No conversions are done for the *msgdata* (input) argument and the *errcode* (input/output) argument because they can contain a mixture of character and binary data. The *ccsid* argument specifies the CCSID for character data in the *msgdata* argument, and users can request CCSID information for the *errcode* argument by using ERRC0200 format. The QMHSNDPM and QMHSNDPM1 i5/OS PASE runtime functions use the current i5/OS PASE CCSID as a default for the *ccsid* value passed to the system API.

See QMHSNDPM()—Send Program Message for further description of the arguments for the QMHSNDPM, QMHSNDPM1, and QMHSNDPM2 i5/OS PASE runtime functions.

Authorities

See QMHSNDPM()—Send Program Message for information about authorities required for the QMHSNDPM, QMHSNDPM1, and QMHSNDPM2 i5/OS PASE runtime functions.

Return Value

The function result is zero for normal completion. The result is nonzero if any input character string could not be converted to the job default CCSID or was too long for the QMHSNDPM API, or if the QMHSNDPM API returned error information in the *errcode* argument.

Usage Notes

1. The system only creates program call message queues ILE procedures and OMI programs, so you cannot send to or receive from a program message queue for a specific function in an i5/OS PASE program.
2. When "*" is specified for the *pgmq* argument, the system locates the program call message queue for an (internal) ILE procedure in service program QP2USER that is the apparent caller of any ILE procedure called by the i5/OS PASE program using i5/OS PASE runtime function _ILECALLX or _ILECALL. i5/OS PASE programs should generally use "*PGMBDY" or "*CTLBDY" instead of "*" to send messages to their caller because a variable number of program call message queues can exist between the queue identified by *pgmq* "*" and the queue for the ILE API that called the i5/OS PASE program.

Related Information

- QMHSNDPM()—Send Program Message (system API)
- "QMHRCVPM()—Receive Program Message for i5/OS PASE" on page 33
- "QMHSNDM()—Send Nonprogram Message for i5/OS PASE" on page 35
- "QMHRCVM()—Receive Nonprogram Message for i5/OS PASE" on page 32

API introduced: V5R1

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Qp2setenv_ile()—Set ILE environment variables for i5/OS PASE

Syntax

```
#include <as400_protos.h>

int Qp2setenv_ile(const char *const *env,
                  const char       *conflict);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **Qp2setenv_ile()** function sets one or more ILE environment variables, with special support to resolve conflicts between ILE and i5/OS PASE variables that have the same name but require different values.

Parameters

env
(Input) Address of a list of pointers to null-terminated character strings that specify ILE environment variables to set. Each character string should have the form ″NAME=value″, where NAME is the environment variable name. The first null pointer indicates the end of the list. ILE environment variables are stored in EBCDIC, so the system converts the character strings from the (ASCII) i5/OS PASE CCSID to the job default CCSID.

conflict
(Input) Pointer to a character string that specifies a colon-delimited list of environment variable names that have conflicting use between i5/OS PASE and ILE. If *conflict* is a null pointer, the system uses a default string of ″SHELL:PATH:LANG:NLSPATH″.

Authorities

None

Return Value

The function result is zero for normal completion. A result of -1 indicates an error that is further qualified by an errno value.

Error Conditions

At least these errno values can be returned, with other values also possible (such as i5/OS-unique ILE errno EAPAR):

[EINVAL]	Input arguments were changed during processing in a way that does not allow the function to complete normally.
[ENOMEM]	Insufficient i5/OS PASE heap memory is available to complete the request.

Usage Notes

1. **Qp2setenv_ILE** sets an ILE environment variable with the same name as the value specified in the *env* string in most cases, but the system adds a prefix ″PASE_″ to the name of the ILE version of some environment variables. The *conflict* argument controls what variables add the name prefix, which lets you pass the current i5/OS PASE environment (runtime variable *environ*) to **Qp2setenv_ile** without removing or changing variables that have conflicting use between i5/OS PASE and ILE. You can specify the address of a null string for the *conflict* argument to avoid any conflict-resolution processing.
2. Any i5/OS PASE environment variable name with a prefix ″ILE_″ is copied to the ILE environment twice. The first copy uses the same variable name, and the second copy uses the name without the prefix. For example, if the i5/OS PASE environment contains a variable named ILE_PATH, the value of this variable is used to set both ILE_PATH and PATH in the ILE environment. This lets you store ILE environment variable values in the i5/OS PASE environment without conflict.

Related Information

- "_PGMCALL()—Call an i5/OS Program for i5/OS PASE" on page 57
- "systemCL()—Run a CL Command for i5/OS PASE" on page 42

API introduced: V5R3

Top | "i5/OS PASE APIs," on page 1 | APIs by category

size_ILEarglist()—Compute ILE Argument List Size for i5/OS PASE

Syntax

```
#include <as400_protos.h>

size_t size_ILEarglist(const arg_type_t  *signature);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **size_ILEarglist()** function computes the number of bytes of memory required to build an ILE argument list for a specific function signature.

Parameters

signature
(Input) Pointer to a list of arg_type_t values that specify the sequence and type of arguments passed to the ILE procedure. ILE procedures can accept a maximum of 400 arguments. The actual number of arguments processed by the **size_ILEarglist** function is determined by the number of entries in the signature list, which is determined by the location of the first ARG_END value in the list. The following values are supported in the signature list:

ARG_END (0)	Specifies the end of the signature list.
ARG_INT8 (-1)	Signed 1-byte integer argument.
ARG_UINT8 (-2)	Unsigned 1-byte integer argument.
ARG_INT16 (-3)	Signed 2-byte integer argument.
ARG_UINT16 (-4)	Unsigned 2-byte integer argument.
ARG_INT32 (-5)	Signed 4-byte integer argument.
ARG_UINT32 (-6)	Unsigned 4-byte integer argument.
ARG_INT64 (-7)	Signed 8-byte integer argument.
ARG_UINT64 (-8)	Unsigned 8-byte integer argument.
ARG_FLOAT32 (-9)	4-byte floating-point argument.
ARG_FLOAT64 (-10)	8-byte floating-point argument.
ARG_MEMPTR (-11)	The argument is a field of type ILEpointer.
ARG_SPCPTR (-12)	The argument is a field of type ILEpointer.
ARG_OPENPTR (-13)	The argument is a field of type ILEpointer.
ARG_MEMTS64 (-14)	The argument is a field of type ts64_t.
ARG_TS64PTR (-15)	The argument is a field of type ts64_t.
Any positive number (1-32767)	The argument is an aggregate (structure or union). The value in the signature list is the length, in bytes, of the aggregate.

Authorities

size_ILEarglist requires no authority.

Return Value

size_ILEarglist returns the number of bytes required to build the ILE argument list (including storage for the ILEarglist_base type and any necessary bytes skipped for alignment between arguments), or zero if an error was detected in the signature list.

Related Information

- "_ILECALLX()—Call an ILE Procedure for i5/OS PASE" on page 49
- "build_ILEarglist()—Build an ILE Argument List for i5/OS PASE" on page 27

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

SQLOverrideCCSID400()—Override SQL CLI CCSID for i5/OS PASE

Syntax

```
#include <as400_protos.h>

int SQLOverrideCCSID400(int newCCSID);
```

Default Public Authority: *USE
Library: i5/OS PASE SQL CLI Library (libdb400.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **SQLOverrideCCSID400()** function allows an i5/OS PASE program to specify a Coded Character Set Identifier (CCSID) used to convert character data arguments and results on i5/OS PASE SQL Call Level Interface (CLI) functions.

Parameters

newCCSID
(Input) Specifies the CCSID used for i5/OS PASE SQL CLI functions.

Authorities

No special authorities required.

Return Value

The function result is zero for success, or -1 for an error that is further qualified by an errno value.

Error Conditions

At least these errno values can be returned:

[EINVAL]	The conversion between newCCSID and the i5/OS job default CCSID is not supported.
[ENFILE]	A converter could not be opened because the maximum number of files in the system are already opened.

[EMFILE]	A converter could not be opened because the maximum number of files are already opened.

Usage Notes

1. The system automatically converts character arguments and results between the CCSID of the job or database field and a CCSID used for i5/OS PASE SQL CLI functions that defaults to the i5/OS PASE CCSID value in effect when the first i5/OS PASE SQL CLI function is called. You must call SQLOverrideCCSID400 before any other i5/OS PASE SQL CLI function, or it will have no effect on CCSID conversions.

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

systemCL()—Run a CL Command for i5/OS PASE

Syntax

```
#include <as400_protos.h>

int systemCL(const char  *command,
             int         flags);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Conditional. See "Usage Notes" on page 43.

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **systemCL()** function runs a CL command.

Parameters

command
(Input) Pointer to a null-terminated string in the i5/OS PASE CCSID that specifies the CL command with any parameters.

flags (Input) Specifies option flags that control how the CL command runs. flags is a bit-wise OR of any of the following values:

SYSTEMCL_MSG_STDOUT (0x00000001)	Directs the system to receive i5/OS messages after normal command completion, convert the text of each message from the job default CCSID to the i5/OS PASE CCSID, and write converted text lines to Integrated File System descriptor 1 (stdout).
SYSTEMCL_MSG_STDERR (0x00000002)	Directs the system to receive i5/OS messages after error command completion, convert the text of each message from the job default CCSID to the i5/OS PASE CCSID, and write converted text lines to Integrated File System descriptor 2 (stderr).
SYSTEMCL_MSG_NOMSGID (0x00000004)	Suppresses message identifiers in text lines written to stdout or stderr for messages processed on behalf of SYSTEMCL_MSG_STDOUT and SYSTEMCL_MSG_STDERR. When this option is omitted, message text lines have the form ″XXX1234: message text″, where ″XXX1234″ is the i5/OS message identifier.

SYSTEMCL_SPOOL_STDOUT (0x00000008)	Directs the system to process any spooled output files created by the CL command by reading each file, converting file data from the job default CCSID to the i5/OS PASE CCSID, and writing converted text lines to Integrated File System descriptor 1 (stdout).
SYSTEMCL_SPOOL_KEEP (0x00000010)	Directs the system to keep any spooled output files after they are processed for option SYSTEMCL_SPOOL_STDOUT, instead of deleting the files after their contents is written to stdout.
SYSTEMCL_FILTER_STDIN (0x00000020)	Directs the system to setup a filter thread that converts from the i5/OS PASE CCSID to the job default CCSID for any data the CL command reads from Integrated File System descriptor 0 (stdin).
SYSTEMCL_FILTER_STDOUT (0x00000040)	Directs the system to setup a filter thread that converts any data the CL command writes to Integrated File System descriptor 1 (stdout) from the job default CCSID to the i5/OS PASE CCSID.
SYSTEMCL_FILTER_STDERR (0x00000080)	Directs the system to setup a filter thread that converts any data the CL command writes to Integrated File System descriptor 2 (stderr) from the job default CCSID to the i5/OS PASE CCSID.
SYSTEMCL_SPAWN (0x00000100)	Directs the system to run the CL command in a separate process. If this option is omitted, the CL command runs in the process that calls the systemCL function.
SYSTEMCL_SPAWN_JOBLOG (0x00000200)	Forces the system to generate an i5/OS job log for the job submitted using option SYSTEMCL_SPAWN.
SYSTEMCL_ENVIRON (0x00000400)	Directs the system to copy i5/OS PASE environment variables to ILE environment variables before running the CL command. This option sets ILE environment variables in the process that calls the systemCL function, regardless of whether the command runs in this process or a child process (created for option SYSTEMCL_SPAWN).

Authorities

No authority is needed to run the **systemCL** function, but the caller must be authorized to run the specified CL command.

Return Value

If the command argument is a null pointer, the function result is zero if system support to call the i5/OS Command Analyzer is available, or a nonzero value otherwise.

If option **SYSTEMCL_SPAWN** is specified, the function result is the exit code from the spawned job (returned by the ILE **waitpid** function), which is non-zero if any error occurred.

Otherwise, the function result is zero for normal command completion, or -1 if an error occurred. No errno value is set for CL command errors.

Usage Notes

1. **systemCL** is only threadsafe in these two cases:
   - You use option **SYSTEMCL_SPAWN** and do not use **SYSTEMCL_ENVIRON**.
   - You only run threadsafe CL commands and do not use **SYSTEMCL_SPAWN**, **SYSTEMCL_FILTER_STDIN**, **SYSTEMCL_FILTER_STDOUT**, **SYSTEMCL_FILTER_STDERR**, or **SYSTEMCL_ENVIRON**.
2. You must set ILE environment variable **QIBM_USE_DESCRIPTOR_STDIO** to Y or I before the CL command does any file I/O to stdin, stdout, or stderr if you need CCSID conversion controlled by options **SYSTEMCL_FILTER_STDIN**, **SYSTEMCL_FILTER_STDOUT**, and **SYSTEMCL_FILTER_STDERR**.
3. Processing for options **SYSTEMCL_FILTER_STDIN**, **SYSTEMCL_FILTER_STDOUT**, and **SYSTEMCL_FILTER_STDERR** creates ILE pthreads (not i5/OS PASE threads) for CCSID conversion

in the process that calls the **systemCL** function. Integrated File System descriptors 0, 1, and 2 are replaced in whatever job runs the CL command with pipes handled by the filter threads. The original file descriptors are restored and the filter threads are ended before the **systemCL** function returns.

4. Many CL commands are not supported in a job with multiple threads. Processing for **SYSTEMCL_SPAWN** runs the CL command in a job that is not multithread-capable, so it can run commands that do not work in a job that is multithread-capable.
5. Processing for option **SYSTEMCL_SPAWN** uses the ILE **spawn** API to run a batch job that inherits ILE attributes such as Integrated File System descriptors and job CCSID, but the batch job does not inherit any i5/OS PASE program (unlike a job created by the i5/OS PASE **fork** function).
6. Processing for **SYSTEMCL_ENVIRON** uses the same name for the ILE copy and the i5/OS PASE environment variable for most variables, but the system adds a prefix ″PASE_″ to the name of the ILE copy of some environment variables. You can control what variables names add the prefix by storing a colon-delimited list of variable names in i5/OS PASE environment variable **PASE_ENVIRON_CONFLICT**. If **PASE_ENVIRON_CONFLICT** is not defined, the system defaults to adding the prefix when copying i5/OS PASE environment variables **SHELL**, **PATH**, **NLSPATH**, and **LANG**.
7. Processing for **SYSTEMCL_ENVIRON** sets two ILE environment variables for each i5/OS PASE environment variable with a name prefix of ″ILE_″. The i5/OS PASE environment variable value is used to set both a variable with the same name and a variable with the name minus the prefix ″ILE_″ in the ILE environment. For example, if the i5/OS PASE environment contains a variable named **ILE_PATH**, the value of this variable is used to set both **ILE_PATH** and **PATH** in the ILE environment.

API introduced: V4R5

Top | “i5/OS PASE APIs,” on page 1 | APIs by category

_CVTERRNO()—Convert ILE errno to i5/OS PASE errno

Syntax

```
#include <as400_protos.h>

int _CVTERRNO(int  errno_ile);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information.

The **_CVTERRNO()** function converts an ILE errno value to a corresponding i5/OS PASE errno value.

Parameters

errno_ile
(Input) Specifies the ILE errno value to convert to a corresponding i5/OS PASE errno value. ILE and i5/OS PASE errno values correspond if they have the same name (for example, EFAULT) in a system-provided header file.

Authorities

_CVTERRNO requires no authority.

Return Value

_CVTERRNO returns the i5/OS PASE equivalent of the input ILE errno value. If the input has no i5/OS PASE errno equivalent (for example, EAPAR is an ILE errno value with no i5/OS PASE equivalent), the input is returned unchanged.

Usage Notes

1. The errno value set by an ILE runtime function must be determined by code running in the same thread and activation group that called the runtime function because ILE runtime sometimes maintains a separate errno variable for each activation group.

Related Information

- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20—Run an i5/OS PASE Program

API introduced: V5R1

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_CVTSPP()—Convert Space Pointer for i5/OS PASE

Syntax

```
#include <as400_protos.h>

void* _CVTSPP(const ILEpointer  *source);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information.

The **_CVTSPP()** function converts the teraspace address in a tagged space pointer to an equivalent i5/OS PASE memory address.

Parameters

source	(Input) Pointer to a tagged space pointer or 16-byte null pointer. The source address must 16-byte aligned.

Authorities

_CVTSPP requires no authority.

Return Value

_CVTSPP returns the i5/OS PASE memory address equivalent of the input tagged space pointer. The result is zero (null) if the input is a 16-byte null pointer or a tagged space pointer that does not contain the teraspace address equivalent of some valid i5/OS PASE memory address.

Error Conditions

Any error is reported with an i5/OS exception message that the system converts to an i5/OS PASE signal. See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

Usage Notes

1. _CVTSPP returns an i5/OS PASE memory address regardless of whether there is currently any memory at that address (as long as the input tagged pointer contains the teraspace address equivalent of a valid i5/OS PASE memory address).

Related Information

- "_CVTTS64()—Convert Teraspace Address for i5/OS PASE"
- "_GETTS64() and _GETTS64_SPP()—Get Teraspace Address for i5/OS PASE" on page 47
- "_GETTS64M()—Get Multiple Teraspace Pointers for i5/OS PASE" on page 48
- "_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE" on page 64
- "_SETSPPM()—Set Multiple Space Pointers for i5/OS PASE" on page 65

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_CVTTS64()—Convert Teraspace Address for i5/OS PASE

Syntax

```
#include <as400_protos.h>

void* _CVTTS64(ts64_t source);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information.

The **_CVTTS64()** function converts a 64-bit teraspace address to an equivalent i5/OS PASE memory address.

Parameters

source (Input) A 64-bit teraspace address.

Authorities

_CVTTS64 requires no authority.

Return Value

_CVTTS64 returns the i5/OS PASE memory address equivalent of the 64-bit teraspace address. The result is zero (null) if the input is either zero or an address that cannot contain i5/OS PASE memory.

Error Conditions

Any error is reported with an i5/OS exception message that the system converts to an i5/OS PASE signal. See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

Usage Notes

1. _CVTTS64 returns an i5/OS PASE memory address regardless of whether there is currently any memory at that address.

Related Information

- "_CVTSPP()—Convert Space Pointer for i5/OS PASE" on page 45
- "_GETTS64() and _GETTS64_SPP()—Get Teraspace Address for i5/OS PASE"
- "_GETTS64M()—Get Multiple Teraspace Pointers for i5/OS PASE" on page 48
- "_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE" on page 64
- "_SETSPPM()—Set Multiple Space Pointers for i5/OS PASE" on page 65

API introduced: V5R3

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_GETTS64() and _GETTS64_SPP()—Get Teraspace Address for i5/OS PASE

Syntax

```
#include <as400_protos.h>

ts64_t _GETTS64(const void  *memory);

ts64_t _GETTS64_SPP(const ILEpointer  *source);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The _GETTS64() function returns the 64-bit teraspace address equivalent of an i5/OS PASE memory address. The _GETTS64_SPP() function returns the 64-bit teraspace address stored in a 16-byte space pointer.

Parameters

memory
(Input) Pointer containing either an i5/OS PASE memory address, or a null pointer (zero).

source (Input) Pointer to a 16-byte tagged space pointer or 16-byte null pointer.

Authorities

_GETTS64 and **_GETTS64_SPP** require no authority.

Return Value

_GETTS64 and **_GETTS64_SPP** return a 64-bit teraspace address or zero.

Error Conditions

Any error is reported with an i5/OS exception message that the system converts to an i5/OS PASE signal. See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

Usage Notes

1. **_GETTS64** returns zero if the input *memory* address is null (zero) or points to a location that cannot contain i5/OS PASE memory.
2. **_GETTS64_SPP** returns zero if *source* is not a tagged space pointer or contains an address that is outside teraspace.
3. **_GETTS64** and **_GETTS64_TS64** return a teraspace address regardless of whether there is currently any memory at the result location.

Related Information

- "_CVTSPP()—Convert Space Pointer for i5/OS PASE" on page 45
- "_CVTTS64()—Convert Teraspace Address for i5/OS PASE" on page 46
- "_GETTS64M()—Get Multiple Teraspace Pointers for i5/OS PASE"
- "_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE" on page 64
- "_SETSPPM()—Set Multiple Space Pointers for i5/OS PASE" on page 65

API introduced: V5R3

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_GETTS64M()—Get Multiple Teraspace Pointers for i5/OS PASE

Syntax

```
#include <as400_protos.h>

void _GETTS64M(ts64_t    *list,
               unsigned  count);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The _GETTS64M() function retrieves teraspace address equivalents for a set of i5/OS PASE memory addresses.

Parameters

list	(Input/Output) Pointer to an array of type ts64_t into which the caller has stored i5/OS PASE memory addresses. _GETTS64M replaces each i5/OS PASE memory address with an equivalent 64-bit teraspace address.
count	(Input) Specifies the number of entries in the *list* array.

Authorities

_GETTS64M requires no authority.

Return Value

_GETTS64M returns no function result.

Error Conditions

Any error is reported with an i5/OS exception message that the system converts to an i5/OS PASE signal. See “i5/OS PASE Signal Handling” on page 82 for information about handling i5/OS exceptions.

Usage Notes

1. **_GETTS64M** returns null (zero) for any input address that is either zero or cannot contain i5/OS PASE memory. i5/OS PASE memory is allocated from teraspace, but teraspace has a limited capacity smaller than 64-bits, so i5/OS PASE can only provide addressability to a subset of a 64-bit address space.
2. **_GETTS64M** returns (non-null) teraspace pointers regardless of whether there is currently any memory at the i5/OS PASE addresses.

Related Information

- “_CVTSPP()—Convert Space Pointer for i5/OS PASE” on page 45
- “_CVTTS64()—Convert Teraspace Address for i5/OS PASE” on page 46
- “_GETTS64() and _GETTS64_SPP()—Get Teraspace Address for i5/OS PASE” on page 47
- “_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE” on page 64
- “_GETTS64M()—Get Multiple Teraspace Pointers for i5/OS PASE” on page 48

API introduced: V5R3

Top | “i5/OS PASE APIs,” on page 1 | APIs by category

_ILECALLX()—Call an ILE Procedure for i5/OS PASE

Syntax

```
#include <as400_protos.h>


int _ILECALLX(const ILEpointer  *target,
              ILEarglist_base   *ILEarglist,
              const arg_type_t  *signature,
              result_type_t     result_type,
              int               flags);


int _ILECALL(const ILEpointer  *target,
             ILEarglist_base   *ILEarglist,
             const arg_type_t  *signature,
             result_type_t     result_type);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information.

The **_ILECALLX()** and **_ILECALL()** functions call an ILE procedure from an i5/OS PASE program. They transfer control to an ILE procedure specified by a 16-byte tagged ILE procedure pointer, passing arguments and returning the function result.

Parameters

target (Input) Pointer to a tagged procedure pointer that addresses the ILE procedure to call. target must be a 16-byte aligned i5/OS PASE memory address.

ILEarglist
(Input/Output) Pointer to a 16-byte aligned ILE argument list structure. ILEarglist is the address of the structure that contains any argument values to pass to the ILE procedure, as well as memory for a function result returned by the ILE procedure. ILEarglist must be long enough to contain all arguments required by the target ILE procedure to avoid unpredictable results.

The base structure of an ILE argument list (including a function result area) is specified by type ILEarglist_base. Any argument values for the ILE procedure are stored in memory immediately following the ILEarglist_base type. The specific format of the argument list is determined by the list of arg_type_t values addressed by the signature argument. The alignment requirements for each argument value in the ILE argument list depends on its length:

Argument Length	Alignment
1 byte	any
2 bytes	2 bytes
3-4 bytes	4 bytes
5-8 bytes	8 bytes
9 or more bytes	16 bytes

signature
(Input) Pointer to a list of arg_type_t values that specify the sequence and type of arguments passed to the ILE procedure. ILE procedures can accept a maximum of 400 arguments. The actual number of arguments processed by the _ILECALLX or _ILECALL function is determined by the number of entries in the signature list, which is determined by the location of the first ARG_END value in the list. The following values are supported in the signature list:

ARG_END (0)	Specifies the end of the signature list.
ARG_INT8 (-1)	Signed 1-byte integer argument.
ARG_UINT8 (-2)	Unsigned 1-byte integer argument.
ARG_INT16 (-3)	Signed 2-byte integer argument.
ARG_UINT16 (-4)	Unsigned 2-byte integer argument.
ARG_INT32 (-5)	Signed 4-byte integer argument.
ARG_UINT32 (-6)	Unsigned 4-byte integer argument.
ARG_INT64 (-7)	Signed 8-byte integer argument.
ARG_UINT64 (-8)	Unsigned 8-byte integer argument.
ARG_FLOAT32 (-9)	4-byte floating-point argument.
ARG_FLOAT64 (-10)	8-byte floating-point argument.
ARG_MEMPTR (-11)	The argument is a field of type ILEpointer into which the caller has stored an i5/OS PASE memory address (in member address). _ILECALLX and _ILECALL convert the i5/OS PASE memory address to an equivalent teraspace address, except that address zero is converted to a special value for a null pointer. The converted result is passed as the argument value to the target ILE procedure. Both functions may update the ILEpointer value so it contains a tagged space pointer.
ARG_SPCPTR (-12)	The argument is a field of type ILEpointer where the i5/OS PASE program has stored a tagged space pointer (or an untagged or null pointer).
» *ARG_SPCPTRI (-16)*	The argument is a field of type ILEpointer into which the caller has stored the i5/OS PASE memory address (in member address) of a tagged space pointer (or an untagged or null pointer) that is passed to the target ILE procedure. The ILEpointer in the argument list may be overlayed with a copy of the space pointer. «
ARG_OPENPTR (-13)	The argument is a field of type ILEpointer where the i5/OS PASE program has stored a 16-byte pointer of any type (including possibly an untagged or null pointer).

» *ARG_OPENPTRI (-17)*	The argument is a field of type ILEpointer into which the caller has stored the i5/OS PASE memory address (in member address) of a 16-byte pointer of any type (including possibly an untagged or null pointer) that is copied into the argument list (overlaying the original ILEpointer value) and passed as the argument to the target ILE procedure. «
ARG_MEMTS64 (-14)	The argument is a field of type ts64_t into which the caller has stored an i5/OS PASE memory address. _ILECALLX and _ILECALL convert the i5/OS PASE memory address to an equivalent 64-bit teraspace pointer, except that null (address zero) is unchanged. The converted result is passed as the argument value to the target ILE procedure. The ts64_t value in the argument list may be overlayed with the converted address.
ARG_TS64PTR (-15)	The argument is a field of type ts64_t where the i5/OS PASE program has stored a 64-bit teraspace address.
Any positive number (1-32767)	The argument is an aggregate (structure or union). The value in the signature list is the length, in bytes, of the aggregate.

result_type
(Input) Specifies the type of function result returned by the ILE procedure.

The following values are supported:

RESULT_VOID(0)	No function result.
RESULT_INT8 (-1)	Signed 1-byte integer result, returned in field result.s_int8.r_int8 in the ILEarglist argument.
RESULT_UINT8 (-2)	Unsigned 1-byte integer result, returned in field result.s_uint8.r_uint8 in the ILEarglist argument.
RESULT_INT16 (-3)	Signed 2-byte integer result, returned in field result.s_int16.r_int16 in the ILEarglist argument.
RESULT_UINT16 (-4)	Unsigned 2-byte integer result, returned in field result.s_uint16.r_uint16 in the ILEarglist argument.
RESULT_INT32 (-5)	Signed 4-byte integer result, returned in field result.s_int32.r_int32 in the ILEarglist argument.
RESULT_UINT32 (-6)	Unsigned 4-byte integer result, returned in field result.s_uint32.r_uint32 in the ILEarglist argument.
RESULT_INT64 (-7)	Signed 8-byte integer result, returned in field result.r_int64 in the ILEarglist argument.
RESULT_UINT64 (-8)	Unsigned 8-byte integer result, returned in field result.r_uint64 in the ILEarglist argument.
RESULT_FLOAT64 (-10)	8-byte floating-point result, returned in field result.r_float64 in the ILEarglist argument.
Any positive number (1-32767)	The function result is an aggregate (structure or union). result_type is the length, in bytes, of the aggregate. An aggregate function result is returned in a buffer allocated by the caller and passed to the target ILE procedure using a special field in the argument list. The caller must provide a buffer large enough for the result returned by the target ILE procedure to avoid unpredictable results. An i5/OS PASE program must set field result.r_aggregate.addr in type ILEarglist_base to the i5/OS PASE memory address of the result buffer before calling an ILE procedure that returns an aggregate result. _ILECALLX and _ILECALL convert the i5/OS PASE memory address to a teraspace address the same way they convert ARG_MEMPTR arguments.

flags (Input) Specifies options to control how the ILE procedure program is called. The flags argument is a bitwise logical-or of one or more of the following values:

ILECALL_NOINTERRUPT (0x00000004)	Specifies that i5/OS PASE signals will not interrupt the called ILE procedure. Some system functions (such as select) can be interrupted by signals. Normally either an ILE signal or an i5/OS PASE signal can interrupt such an operation, but ILECALL_NOINTERRUPT delays i5/OS PASE signal processing until control returns from the called ILE procedure. This option has no effect on ILE signal handling.

Authorities

_ILECALL and **_ILECALLX** require no authority.

Return Value

Most errors from **_ILECALLX** and **_ILECALL** are reported with i5/OS exception messages that are converted to i5/OS PASE signals. See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

If no i5/OS PASE signal is sent, one of these values is returned:

ILECALL_NOERROR(0)	The target ILE procedure was called and returned normally.
ILECALL_INVALID_ARG (1)	An invalid value was found in the signature list.
ILECALL_INVALID_RESULT (2)	The result_type value is invalid.
ILECALL_INVALID_FLAGS (3)	The flags value is invalid.

Usage Notes

1. **_ILECALLX** and **_ILECALL** can only call ILE procedures in an i5/OS bound program. If your i5/OS PASE program needs to call an i5/OS program object (object type *PGM), you must use the **_PGMCALL** function or use the **systemCL** function to run the CL CALL command.
2. Every module in a *PGM or *SRVPGM object containing a function directly called by PASE (using **_ILECALLX** or **_ILECALL**) must be Teraspace-safe. If any module in the program was created as TERASPACE(*NO), then i5/OS PASE will not be able to call any procedure in that program (even a procedure in a module created as TERASPACE(*YES)).
3. **_ILECALLX** and **_ILECALL** do no character encoding conversions, so the i5/OS PASE program may need to convert argument and result character strings between ASCII and EBCDIC. i5/OS PASE runtime function **iconv** can be used for character conversions.
4. An i5/OS PASE program can pass tagged space pointer arguments to an ILE procedure using either ARG_SPCPTR or ARG_OPENPTR unless the target ILE procedure uses ARGOPT linkage, in which case ARG_SPCPTR must be used. ARG_MEMPTR can be used for space pointer arguments regardless of what linkage is used by the target ILE procedure.
5. ILE procedure pointers address resources inside an ILE activation group. The machine prohibits use of activation group resources from a process other than the owner of the activation group. This means that the child process of a **fork** cannot use ILE procedure pointers inherited from the parent process. The child process can, however, use **_ILELOADX** to load the bound program (creating a new activation in the child process) and then use **_ILESYMX** to obtain ILE procedure pointers into the new activation.
6. See "_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE" on page 64 (_SETSPP) for more information about tagged space pointers and sharing tagged pointers between processes.
7. **_ILECALL** is equivalent to **_ILECALLX** with the **ILECALL_NOINTERRUPT** flag.

Related Information

- "_PGMCALL()—Call an i5/OS Program for i5/OS PASE" on page 57
- "_ILESYMX()—Find an Exported ILE Symbol for i5/OS PASE" on page 54
- "_ILELOADX()—Load an ILE Bound Program for i5/OS PASE" on page 53
- "size_ILEarglist()—Compute ILE Argument List Size for i5/OS PASE" on page 40
- "build_ILEarglist()—Build an ILE Argument List for i5/OS PASE" on page 27

- “_CVTSPP()—Convert Space Pointer for i5/OS PASE” on page 45
- “_CVTTS64()—Convert Teraspace Address for i5/OS PASE” on page 46
- “_GETTS64() and _GETTS64_SPP()—Get Teraspace Address for i5/OS PASE” on page 47
- “_GETTS64M()—Get Multiple Teraspace Pointers for i5/OS PASE” on page 48
- “_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE” on page 64
- “_SETSPPM()—Set Multiple Space Pointers for i5/OS PASE” on page 65
- “systemCL()—Run a CL Command for i5/OS PASE” on page 42

API introduced: V4R5

Top | “i5/OS PASE APIs,” on page 1 | APIs by category

_ILELOADX()—Load an ILE Bound Program for i5/OS PASE

Syntax

```
#include <as400_protos.h>


unsigned long long _ILELOADX(const void  *id,
                             unsigned int flags);


int _ILELOAD(const void  *id,
             unsigned int flags);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **_ILELOADX()** and **_ILELOAD()** functions load a bound program into the ILE activation group associated with the procedure that started i5/OS PASE (either the activation group that called the Qp2RunPase API, or the default activation group for a job running program QP2FORK).

Parameters

id (Input) Pointer to the identification of the bound program.id is either the address of a null-terminated character string in the i5/OS PASE CCSID that names the program, or the address of a system pointer to the program, depending on the value of the flags argument.

flags (Input) Specifies options to control how the bound program is found and activated. The flags argument is a bitwise logical-or of one or more of the following values:

ILELOAD_PATH (0x00000000)	Specifies that the id argument is the address of a string that contains an absolute or relative path in the Integrated File System to a program or service program object. Alphabetic case is either ignored or honored depending on the attributes of the File System that contains the path. ILELOAD_PATH, ILELOAD_LIBOBJ, and ILELOAD_PGMPTR are mutually exclusive.
ILELOAD_LIBOBJ (0x00000001)	Specifies that the id argument is the address of a string that contains a qualified library/object name of a service program (where omitting the library name implies resolving to the object through the job library list). Alphabetic case is honored when searching for a library/object name (so the string should be all uppercase). ILELOAD_PATH, ILELOAD_LIBOBJ, and ILELOAD_PGMPTR are mutually exclusive.

ILELOAD_PGMPTR (0x00000002)	Specifies that the id argument is the address of a system pointer to the bound program (object type *SRVPGM or *PGM) to load. ILELOAD_PATH, ILELOAD_LIBOBJ, and ILELOAD_PGMPTR are mutually exclusive.

Authorities

_ILELOADX and **_ILELOAD** call the ILE **QleActBndPgmLong** API to activate the bound program. See QleActBndPgmLong()—Activate Bound Program Long for information about authorities required to use **_ILELOADX** and **_ILELOAD**.

Return Value

A function result of -1 indicates an error that is further qualified by an errno value. If the bound program was successfully activated (including the case where it was already activated before _ILELOADX or _ILELOAD ran), the function result is an activation mark that uniquely identifies the activation within the process. 64-bit ILE activation mark values can only be returned using _ILELOADX.

Error Conditions

Memory errors and errors while activating the bound program may be reported with an i5/OS exception message that the system converts to an i5/OS PASE signal (not return code and errno values). See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

At least these errno values can be returned, with other values also possible (such as i5/OS-unique ILE errno EAPAR):

[EACCES]	Not authorized to a library or directory needed to resolve the id.
[EBUSY]	A library or directory needed to resolve the specified id is currently in use (locked).
[EFAULT]	A memory fault occurred attempting to reference the id.
[EINVAL]	An invalid argument value was specified.
[EINTER]	An signal interrupted the operation.
[ENAMETOOLONG]	Some component of the specified id is too long, or the entire id exceeds the system limit.
[ENOENT]	No file/object was found for the specified id.
[ENOTDIR]	A qualifier part of the id is not a directory.
[ELOOP]	Too many levels of symbolic links.

Related Information

- "_ILECALLX()—Call an ILE Procedure for i5/OS PASE" on page 49
- "_ILESYMX()—Find an Exported ILE Symbol for i5/OS PASE"
- "_RSLOBJ()—Resolve to an i5/OS Object for i5/OS PASE" on page 60

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_ILESYMX()—Find an Exported ILE Symbol for i5/OS PASE

Syntax

```
#include <as400_protos.h>
```

```
int _ILESYMX(ILEpointer          *export,
             unsigned long long  actmark,
             const char          *symbol);
```

```
int _ILESYM(ILEpointer  *export,
            int         actmark,
            const char  *symbol);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **_ILESYMX()** and **_ILESYM()** functions find an exported symbol in the activation of an ILE bound program and return a 16-byte tagged pointer to the data or procedure for the symbol.

Parameters

export (Output) Pointer to a 16-byte aligned buffer for the tagged pointer return value. The export buffer used to store a tagged pointer to the data or procedure for the exported symbol.

actmark
(Input) Specifies an activation mark that identifies the activation (in the current i5/OS job) to search for the symbol. A value of zero causes the system to search all activations in the activation group that started i5/OS PASE (either the activation group that called the **Qp2RunPase** API, or the default activation group for a job running program **QP2FORK**). The **_ILELOADX** and **_ILELOAD** functions return an activation mark when they load a bound program. 64-bit activation mark values can only be handled by **_ILESYMX** and **_ILELOADX**.

symbol
(Input) Pointer to the symbol name to find. symbol is the address of a null-terminated character string in the i5/OS PASE CCSID that specifies the name of a symbol exported by the actmark activation.

Authorities

_ILESYMX and **_ILESYM** call the ILE **QleGetExpLong** API to find the exported symbol. See QleGetExpLong()—Get Export Long for information about authorities required to use _ILESYMX and _ILESYM.

Return Value

A function result of -1 indicates an error that is further qualified by an errno value. If the symbol was successfully found, the export pointer is set to the address of the function or data for the symbol, and the function result is set to one of these values:

ILESYM_PROCEDURE (1)	The export return value is a tagged pointer to an ILE procedure. An ILE procedure pointer can be used with the _ILECALLX function to call the ILE procedure.
ILESYM_DATA(2)	The export return value is a tagged space pointer to a data item in the ILE activation.

Error Conditions

Memory errors and errors during ILE symbol resolution processing may be reported with an i5/OS exception message that the system converts to an i5/OS PASE signal (not return code and errno values). See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

At least these errno values can be returned, with other values also possible (such as i5/OS-unique ILE errno EAPAR):

[EACCES]	Not authorized to the actmark activation.
[ENOENT]	The symbol was not found in the actmark activation.

Related Information

- "_ILELOADX()—Load an ILE Bound Program for i5/OS PASE" on page 53
- "_ILECALLX()—Call an ILE Procedure for i5/OS PASE" on page 49

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_MEMCPY_WT()—Copy Memory With Tags for i5/OS PASE

Syntax

```
#include <as400_protos.h>

void* _MEMCPY_WT(void        *target,
                 const void  *source,
                 size_t      length);

void _MEMCPY_WT2(const ILEpointer  *target,
                 const ILEpointer  *source,
                 size_t            length);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The _MEMCPY_WT() and _MEMCPY_WT2() functions copy memory without destroying 16-byte tagged pointers.

Standard memory copy functions such as memcpy never produce a usable tagged pointer in the target memory. _MEMCPY_WT and _MEMCPY_WT2 copy memory in a way that preserves the integrity of any complete (16-byte) tagged pointers copied, as long as the source and target have the same alignment with respect to a 16-byte boundary.

Parameters

target (Output) Pointer to target memory. For _MEMCPY_WT, target is the i5/OS PASE address of the target memory. For _MEMCPY_WT2, target is the 16-byte aligned address of a tagged space pointer to the target memory.

source (Input) Pointer to source memory. For _MEMCPY_WT, source is the i5/OS PASE address of the source memory. For _MEMCPY_WT2, source is the 16-byte aligned address of a tagged space pointer to the source memory.

length (Input) Specifies the number of bytes to copy between the source and target.

Authorities

_MEMCPY_WT and _MEMCPY_WT2 require no authority.

Return Value

_MEMCPY_WT returns the target memory address. _MEMCPY_WT2 returns no function result.

Error Conditions

Any error is reported with an i5/OS exception message that the system converts to an i5/OS PASE signal. See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

Usage Notes

1. _MEMCPY_WT only copies between memory areas in the i5/OS PASE address space. _MEMCPY_WT2 can copy between any memory areas addressable through tagged space pointers, which need not be in the i5/OS PASE address space.
2. Memory is copied without error if the source and target do not have the same alignment with respect to a 16-byte boundary or if only part of a tagged pointer is copied, but the target will not contain a usable tagged pointer.
3. _MEMCPY_WT and _MEMCPY_WT2 are implemented with kernel system calls, so they generally run slower than memcpy.

Related Information

- "_CVTSPP()—Convert Space Pointer for i5/OS PASE" on page 45
- "_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE" on page 64
- "_SETSPPM()—Set Multiple Space Pointers for i5/OS PASE" on page 65

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_PGMCALL()—Call an i5/OS Program for i5/OS PASE

Syntax

```
#include <as400_protos.h>

int _PGMCALL(const ILEpointer  *target,
             void             **argv,
             unsigned          flags);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **_PGMCALL()** function calls an i5/OS program (object type *PGM) from an i5/OS PASE program. It transfers control to the *PGM object specified by a 16-byte tagged system pointer (passing any necessary arguments) and resumes execution when control returns.

Parameters

target (Input) Pointer to a tagged system pointer that addresses the i5/OS program (object type *PGM) to call. *target* must be a 16-byte aligned i5/OS PASE memory address.

argv (Input/Output) Array of pointers to arguments. *argv* is the address of an array of pointers to argument variables that are (usually) passed by-address to the i5/OS program. *argv* can be zero (null) if there are no arguments to pass. The last element in the array must be a null pointer. A maximum of **PGMCALL_MAXARGS** (255) arguments can be passed to an i5/OS program unless **PGMCALL_NOMAXARGS** is specified.

flags (Input) Specifies options to control how the i5/OS program is called. The *flags* argument is a bitwise logical-or of one or more of the following values:

PGMCALL_DIRECT_ARGS (0x00000001)	Specifies that the addresses in the *argv* array are passed directly to the i5/OS program as formal arguments. If **PGMCALL_DIRECT_ARGS** is omitted, the system builds tagged space pointers to the argument memory locations identified in the *argv* array and passes the space pointers as formal arguments (standard i5/OS program linkage).
PGMCALL_DROP_ADOPT (0x00000002)	Specifies that authorizations adopted by i5/OS program invocations already in the stack are dropped so the called program only benefits from authorizations available to the user and group profiles for the thread.
PGMCALL_NOINTERRUPT (0x00000004)	Specifies that i5/OS PASE signals will not interrupt the called i5/OS program. Some system functions (such as **select**) can be interrupted by signals. Normally either an ILE signal or an i5/OS PASE signal can interrupt such an operation, but **PGMCALL_NOINTERRUPT** delays i5/OS PASE signal processing until control returns from the called i5/OS program. This option has no effect on ILE signal handling.
PGMCALL_NOMAXARGS (0x00000008)	Specifies that more than 255 arguments may be passed to the i5/OS program. Current system architecture limits the number of arguments to 16383 when this flag is set. Specifying **PGMCALL_NOMAXARGS** slightly increases system resource requirements even when no more than 255 arguments are provided.
PGMCALL_ASCII_STRINGS (0x00000010)	Specifies that all arguments are null-terminated ASCII character strings. The system converts argument strings from the i5/OS PASE CCSID to the job default CCSID and passes the converted (EBCDIC) strings to the i5/OS program. The converted strings are stored in an internal system buffer, so any changes made by the i5/OS program are **not** returned to the caller (by-value argument behavior).

Authorities

Object Referred to	Authority Required
i5/OS program to call	*X

Return Value

Most errors from **_PGMCALL** are reported with i5/OS exception messages that are converted to i5/OS PASE signals. See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

If no i5/OS PASE signal is sent, a function result of zero indicates the i5/OS program was called and returned normally. A function result of -1 indicates an error that is further qualified by an errno value.

Error Conditions

At least these errno values can be returned, with other values also possible (such as i5/OS-unique ILE errno EAPAR):

[EINVAL]	An invalid *flags* value was specified, or too many arguments were provided.

Usage Notes

1. **_PGMCALL** can only call i5/OS program objects (object type *PGM). If your i5/OS PASE program needs to call a particular ILE procedure inside a *PGM or *SRVPGM object, you must to use the **_ILECALL** function.
2. You can use the **_RSLOBJ** or **_RSLOBJ2** function to obtain a system pointer to an i5/OS program (object type *PGM).
3. Any i5/OS program that accepts arguments must be Teraspace-safe (created using TERASPACE(*YES)) to be called using **_PGMCALL** because the arguments are usually passed in Teraspace storage.
4. Arguments (addressed by pointers in the *argv* array) can be of any data type. Arguments are passed by-address, so the called i5/OS program can modify argument variables to return results to the i5/OS PASE program unless **PGMCALL_ASCII_STRINGS** is specified.
5. **_PGMCALL** does no character encoding conversions unless **PGMCALL_ASCII_STRINGS** is specified,so the i5/OS PASE program may need to convert argument and result character strings between ASCII and EBCDIC. i5/OS PASE runtime function **iconv** can be used for character conversions.

Related Information

- "_RSLOBJ()—Resolve to an i5/OS Object for i5/OS PASE" on page 60
- "_ILECALLX()—Call an ILE Procedure for i5/OS PASE" on page 49
- "_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE" on page 64
- "_SETSPPM()—Set Multiple Space Pointers for i5/OS PASE" on page 65
- "Qp2setenv_ile()—Set ILE environment variables for i5/OS PASE" on page 38

API introduced: V5R2

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_RETURN()—Return Without Exiting i5/OS PASE

Syntax

```
#include <as400_protos.h>

int _RETURN(void);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: No

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The _RETURN() function returns to the ILE caller that called i5/OS PASE in this job, without exiting the i5/OS PASE program. i5/OS PASE remains active in the job, so APIs Qp2CallPase and Qp2CallPase2 can be used to call procedures in the i5/OS PASE program.

Parameters

None.

Authorities

None.

Return Value

_RETURN does not return to the i5/OS PASE program if it successfully returns to the ILE caller. A function result of -1 with an errno is returned for any error.

Error Conditions

EPERM is set if _RETURN is used in a fork child process or in an i5/OS PASE program that is currently running multiple threads.

Usage Notes

1. The system provides two i5/OS PASE programs, /usr/lib/start32 (for 32-bits) and /usr/lib/start64 (for 64-bits), that return without exiting immediately after initializing the standard C library (libc.a) and pthreads library (libpthreads.a).
2. The system ends any i5/OS PASE program when it destroys the activation group that called API Qp2RunPase. API program QP2SHELL always calls Qp2RunPase in a *NEW activation group that is destroyed before return, so QP2SHELL is not useful for running an i5/OS PASE program that returns without exiting.
3. You may need to call ILE API Qp2EndPase to end an i5/OS PASE program that uses **_RETURN**. See "Qp2EndPase()—End an i5/OS PASE Program" on page 15 for more information.

Related Information

- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20—Run an i5/OS PASE Program
- "Qp2EndPase()—End an i5/OS PASE Program" on page 15—End an i5/OS PASE Program

API introduced: V5R2

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_RSLOBJ()—Resolve to an i5/OS Object for i5/OS PASE

Syntax

```
#include <as400_protos.h>

int _RSLOBJ(ILEpointer       *sysptr,
            const char       *path,
            char             *objtype);

int _RSLOBJ2(ILEpointer      *sysptr,
             unsigned short  type_subtype,
             const char      *objname,
             const char      *libname);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **_RSLOBJ()** and **_RSLOBJ2()** functions resolve to an i5/OS object. They accept symbolic information that identifies the object and return a 16-byte tagged system pointer to the specified object.

Parameters

sysptr (Output) Pointer to the i5/OS object. *sysptr* is the address of a 16-byte aligned buffer allocated by the caller and used to return a system pointer to the i5/OS object.

path (Input) Pointer to an Integrated File System path name that locates the i5/OS object. *path* is the address of a null-terminated string in the i5/OS PASE CCSID that contains a path name for the i5/OS object.

objtype
(Output) Pointer to the returned i5/OS object type. *objtype* is the address of a buffer allocated by the caller and used to return a null-terminated string in the i5/OS PASE CCSID that identifies the i5/OS object type. If *objtype* is a null pointer, no i5/OS object type is returned. When *objtype* is not null, the caller must provide a buffer of length **RSLOBJ_OBJTYPE_MAXLEN** (11) to avoid errors.

type_subtype
(Input) Object type and subtype. *type_subtype* specifies the MI object type and MI object subtype of the i5/OS object. Header file <as400_types.h> declares these constants for type and subtype values:

RSLOBJ_TS_PGM (0x0201)	Specifies the MI type and subtype for an i5/OS program (object type *PGM).
RSLOBJ_TS_SRVPGM (0x0203)	Specifies the MI type and subtype for an i5/OS service program (object type *SRVPGM).

objname
(Input) Pointer to the name of the i5/OS object. *objname* is the address of a null-terminated string in the i5/OS PASE CCSID that contains the name of the i5/OS object.

libname
(Input) Pointer to the name of the i5/OS library that contains the object. *libname* is the address of a null-terminated string in the i5/OS PASE CCSID that contains the name of an i5/OS library. Specifying a null pointer or a pointer to a null string is the same as specifying "*LIBL", which searches the thread library list.

Authorities

Object Referred to	Authority Required
Every directory in the Integrated File System path to the i5/OS object	*X
i5/OS library that contains the object	*X

Return Value

The function result is zero if the i5/OS object was found and a system pointer was returned in the *sysptr* argument. A function result of -1 indicates an error that is further qualified by an errno value.

Error Conditions

Memory errors may be reported with an i5/OS exception message that the system converts to an i5/OS PASE signal (not return code and errno values). See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

At least these errno values can be returned, with other values also possible (such as i5/OS-unique ILE errno EAPAR):

[EACCES]	Not authorized to a library or directory needed to resolve to the i5/OS object.
[EBUSY]	A library or directory needed to resolve to the i5/OS object is currently in use (locked).
[EFAULT]	A memory fault occurred attempting to reference an argument.
[EINVAL]	An invalid argument value was specified.
[EINTER]	An signal interrupted the operation.
[ENAMETOOLONG]	Some component of the specified *path* is too long, or the entire *path* exceeds the system limit, or the *objname* or *libname* string is longer than 30 characters.
[ENOENT]	The specified i5/OS object was not found.
[ENOTDIR]	A qualifier part of the *path* is not a directory.
[ELOOP]	Too many levels of symbolic links.

Usage Notes

1. For **_RSLOBJ**, alphabetic case is either ignored or honored depending on the attributes of the file system that contains the path. Alphabetic case is always honored by **_RSLOBJ2**, so the *objname* and *libname* strings must be uppercase.

Related Information

- "_ILELOADX()—Load an ILE Bound Program for i5/OS PASE" on page 53
- "_PGMCALL()—Call an i5/OS Program for i5/OS PASE" on page 57

API introduced: V5R2

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_SETCCSID()—Set i5/OS PASE CCSID

Syntax

```
#include <as400_protos.h>

int _SETCCSID(int  ccsid);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: No

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The _SETCCSID() function returns the previous value of the i5/OS PASE Coded Character Set Identifier (CCSID) and optionally sets a new i5/OS PASE CCSID.

Parameters

ccsid	(Input) Specifies the new i5/OS PASE CCSID value, or -1 to retrieve the current i5/OS PASE CCSID without changing it. An i5/OS PASE CCSID must be either a single-byte ASCII encoding that the ILE version of **iconv** can convert to and from the job default CCSID, or 1208 for UTF-8 encoding.

Authorities

_SETCCSID requires no authority.

Return Value

_SETCCSID returns either the original i5/OS PASE CCSID (before it was changed), or -1 if an error occurred and the i5/OS PASE CCSID was left unchanged.

Error Conditions

The only error condition that causes a function result of -1 is that the new *ccsid* cannot be converted to or from the i5/OS job default CCSID.

Usage Notes

1. The initial i5/OS PASE CCSID value is specified as a parameter on the Qp2RunPase API. The i5/OS PASE CCSID has two primary uses:

   - It is used to set the the CCSID attribute of any bytestream file created in the Integrate File System by an i5/OS PASE program.
   - It is used by i5/OS PASE runtime functions to convert character arguments and results between the i5/OS PASE CCSID and whatever encoding is required by the i5/OS service used to implement the function.
2. The i5/OS PASE CCSID should generally be the CCSID equivalent of the code set for the current locale. See "i5/OS PASE Locales" on page 70 to determine what locales are supported by i5/OS PASE.
3. Character arguments and results for i5/OS PASE runtime functions that use i5/OS services are almost always automatically converted using the i5/OS PASE CCSID. For example, the name of a bytestream file passed to the i5/OS PASE open function is converted from the i5/OS PASE CCSID to the internal encoding required by the i5/OS Integrated File System.
4. Any data an i5/OS PASE program writes to or reads from a file descriptor for an open bytestream file, socket, FIFO, or pipe is generally *not* converted. The only exception is for the initial file descriptors 0, 1, and 2 provided when the Qp2RunPase API is called to start an i5/OS PASE program, which default to converting file data between the i5/OS PASE CCSID and the job default CCSID (see "Qp2RunPase()—Run an i5/OS PASE Program" on page 20 (Qp2RunPase) for more information).
5. Other than special support for file descriptors 0, 1, and 2, i5/OS PASE runtime does no CCSID conversion of file data. This differs from ILE runtime, which does CCSID conversion between the file CCSID and job default CCSID for any file opened in text mode. i5/OS PASE runtime sets the CCSID attribute of any file it creates to the i5/OS PASE CCSID, so an ILE program that uses text mode to open an ASCII file created by an i5/OS PASE program can read and write EBCDIC data.
6. The i5/OS PASE runtime functions cstoccsid and ccsidtocs convert between AIX Character Set names and CCSID values.

Related Information

- "Qp2RunPase()—Run an i5/OS PASE Program" on page 20—Run an i5/OS PASE Program

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE

Syntax

```
#include <as400_protos.h>

void _SETSPP(ILEpointer  *target,
             const void  *memory);


void _SETSPP_TS64(ILEpointer  *target,
                  ts64_t      ts64);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: These functions can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The _SETSPP() function sets a tagged space pointer to the teraspace equivalent of an i5/OS PASE memory address. The _SETSPP_TS64() function sets a tagged space pointer to the memory identified by a 64-bit teraspace address.

Parameters

target (Output) Pointer to a 16-byte aligned buffer where the tagged space pointer (or null pointer) is returned.

memory
(Input) Pointer containing either an i5/OS PASE memory address, or a null pointer (zero).

ts64 (Input) 64-bit teraspace address, or null (zero).

Authorities

_SETSPP and **_SETSPP_TS64** require no authority.

Return Value

_SETSPP and **_SETSPP_TS64** return no function result. A tagged space pointer or 16-byte null pointer is returned in the *target* buffer.

Error Conditions

Any error is reported with an i5/OS exception message that the system converts to an i5/OS PASE signal. See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

Usage Notes

1. **_SETSPP** returns a 16-byte null pointer if the input i5/OS PASE *memory* address is null (zero) or if a 64-bit *memory* value points to a location that cannot contain i5/OS PASE memory. i5/OS PASE memory is allocated from teraspace, but teraspace has a limited capacity smaller than 64-bits, so i5/OS PASE can only provide addressability to a subset of a 64-bit address space.
2. **_SETSPP_TS64** returns a 16-byte null pointer if the input *ts64* value is zero or outside the range of teraspace.

3. **_SETSPP** and **_SETSPP_TS64** return *target* a space pointer regardless of whether there is currently any memory at the target address.
4. A tagged space pointer to a teraspace location must only be used by the process that owns the teraspace, although the current system implementation does not reliably enforce this restriction. Applications must not assume that a process can reference memory in the teraspace of another process because future system implementations may make this impossible. Tagged space pointers to teraspace memory that were either inherited by the child process of a fork or stored in shared memory by another process should be considered unusable.
5. Tagged (16-byte) pointers must not be stored in memory mapped from a bytestream file (by either mmap or shmat) although the current system implementation does not reliably enforce this restriction. Tagged pointers can be stored in shared memory objects (created by shmget and mapped by shmat), but a tagged space pointer to teraspace memory cannot be reliably used by a process other than the one that owns the teraspace.

Related Information

- "_CVTSPP()—Convert Space Pointer for i5/OS PASE" on page 45
- "_CVTTS64()—Convert Teraspace Address for i5/OS PASE" on page 46
- "_GETTS64() and _GETTS64_SPP()—Get Teraspace Address for i5/OS PASE" on page 47
- "_GETTS64M()—Get Multiple Teraspace Pointers for i5/OS PASE" on page 48
- "_SETSPPM()—Set Multiple Space Pointers for i5/OS PASE"

API introduced: V4R5 for _SETSPP, V5R3 for _SETSPP_TS64

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_SETSPPM()—Set Multiple Space Pointers for i5/OS PASE

Syntax

```
#include <as400_protos.h>

void _SETSPPM(ILEpointer *const *target);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The _SETSPPM() function sets multiple tagged space pointers to the teraspace equivalents of i5/OS PASE memory addresses.

Parameters

target (Input/Output) Pointer to a list of pointers (of type ILEpointer), with a null pointer marking the end of the list. _SETSPPM updates each ILEpointer with a tagged space pointer to the teraspace equivalent address of the i5/OS PASE memory address input through the *addr* field of the ILEpointer.

Authorities

_SETSPPM requires no authority.

Return Value

_SETSPPM returns no function result.

Error Conditions

Any error is reported with an i5/OS exception message that the system converts to an i5/OS PASE signal. See “i5/OS PASE Signal Handling” on page 82 for information about handling i5/OS exceptions.

Usage Notes

1. **_SETSPPM** returns a 16-byte null ILEpointer if the i5/OS PASE memory address is null (zero) or points to a location that cannot contain i5/OS PASE memory. i5/OS PASE memory is allocated from teraspace, but teraspace has a limited capacity smaller than 64-bits, so i5/OS PASE can only provide addressability to a subset of a 64-bit address space.
2. **_SETSPPM** returns space pointers regardless of whether there is currently any memory at the i5/OS PASE addresses.
3. A tagged space pointer to a teraspace location must only be used by the process that owns the teraspace, although the current system implementation does not reliably enforce this restriction. Applications must not assume that a process can reference memory in the teraspace of another process because future system implementations may make this impossible. Tagged space pointers to teraspace memory that were either inherited by the child process of a fork or stored in shared memory by another process should be considered unusable.
4. Tagged (16-byte) pointers must not be stored in memory mapped from a bytestream file (by either mmap or shmat) although the current system implementation does not reliably enforce this restriction. Tagged pointers can be stored in shared memory objects (created by shmget and mapped by shmat), but a tagged space pointer to teraspace memory cannot be reliably used by a process other than the one that owns the teraspace.

Related Information

- “_CVTSPP()—Convert Space Pointer for i5/OS PASE” on page 45
- “_CVTTS64()—Convert Teraspace Address for i5/OS PASE” on page 46
- “_GETTS64() and _GETTS64_SPP()—Get Teraspace Address for i5/OS PASE” on page 47
- “_GETTS64M()—Get Multiple Teraspace Pointers for i5/OS PASE” on page 48
- “_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE” on page 64

API introduced: V5R3

Top | “i5/OS PASE APIs,” on page 1 | APIs by category

_STRLEN_SPP()—Determine Character String Length for i5/OS PASE

Syntax

```
#include <as400_protos.h>

size_t _STRLEN_SPP(const ILEpointer  *string);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **_STRLEN_SPP()** determines the length of a null-terminated character string. It performs the same operation as the **strlen** function, but uses a 16-byte tagged space pointer to locate the string.

Parameters

string (Input) Pointer to character string. string is the 16-byte aligned address of a tagged space pointer to the character string.

Authorities

_STRLEN_SPP requires no authority.

Return Value

_STRLEN_SPP returns length of the character string.

Error Conditions

Any error is reported with an i5/OS exception message that the system converts to an i5/OS PASE signal. See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

Usage Notes

1. _STRLEN_SPP can reference any memory addressable through a tagged space pointer, which need not be in the i5/OS PASE address space.
2. _STRLEN_SPP is implemented with a kernel system call, so it generally runs slower than strlen.

Related Information

- "_CVTSPP()—Convert Space Pointer for i5/OS PASE" on page 45
- "_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE" on page 64
- "_SETSPPM()—Set Multiple Space Pointers for i5/OS PASE" on page 65

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

_STRNCPY_SPP()—Copy Character String for i5/OS PASE

Syntax

```
#include <as400_protos.h>

void _STRNCPY_SPP(const ILEpointer  *target,
                  const ILEpointer  *source,
                  size_t            length);
```

Default Public Authority: *USE
Library: Standard C Library (libc.a)
Threadsafe: Yes

Note: This function can only be used in an i5/OS PASE program. See i5/OS PASE for more information about creating i5/OS PASE programs.

The **_STRNCPY_SPP()** function copies a null-terminated character string. It performs the same operation as the **strncpy** function, but uses 16-byte tagged space pointers to locate the source and target strings.

Parameters

target (Output) Pointer to target buffer. Target is the 16-byte aligned address of a tagged space pointer to the target buffer.

source (Input) Pointer to source string. source is the 16-byte aligned address of a tagged space pointer to the source character string.

length (Input) Specifies the maximum number of bytes to copy between the source and target. If the source string is too long, then only the specified number of bytes are copied and the target string is not terminated with a null. If the source string is too short, the copy is padded with nulls to fill the target buffer.

Authorities

_STRNCPY_SPP requires no authority.

Error Conditions

Any error is reported with an i5/OS exception message that the system converts to an i5/OS PASE signal. See "i5/OS PASE Signal Handling" on page 82 for information about handling i5/OS exceptions.

Usage Notes

1. **_STRNCPY_SPP** can copy between any memory areas addressable through tagged space pointers, which need not be in the i5/OS PASE address space.
2. **_STRNCPY_SPP** is implemented with a kernel system call, so it generally runs slower than **strncpy**.

Related Information

- "_CVTSPP()—Convert Space Pointer for i5/OS PASE" on page 45
- "_SETSPP() and _SETSPP_TS64()—Set Space Pointer for i5/OS PASE" on page 64
- "_SETSPPM()—Set Multiple Space Pointers for i5/OS PASE" on page 65

API introduced: V4R5

Top | "i5/OS PASE APIs," on page 1 | APIs by category

Concepts

These are the concepts for this category.

i5/OS PASE Runtime Libraries

i5/OS PASE runtime supports a large subset of the interfaces provided by AIX runtime. Most runtime interfaces supported by i5/OS PASE provide the same options and behavior as AIX. The latest information about what AIX runtime interfaces are supported by i5/OS PASE can found at the PartnerWorld for Developers, iSeries web site.

i5/OS PASE interfaces for Structured Query Language (SQL) Call Level Interface (CLI) are somewhat different from any AIX database. i5/OS PASE library **libdb400.a** handles (ASCII/EBCDIC) character encoding conversions, but supports only the options and behaviors provided by DB2 Universal Database for iSeries. An i5/OS PASE program that uses SQL CLI must compile using i5/OS header file **sqlcli.h**. See i5/OS PASE for more information.

i5/OS PASE runtime includes the following libraries, installed (as symbolic links) in /usr/lib. See AIX documentation for information about most of the interfaces exported by these libraries, DB2 Universal Database for iSeries documentation for information about SQL CLI interfaces, and "i5/OS PASE APIs," on page 1 for information about interfaces that are unique to i5/OS PASE:

Library	Description
libbsd.a	BSD UNIX(TM) equivalence runtime
libc.a	C runtime
libC.a	C++ runtime
libc128.a	C 128-bit (type long double) runtime
libC128.a	C++ 128-bit (type long double) runtime
libcrypt.a	C runtime cryptographic interfaces
libcur.a	AIX legacy Curses library
libdb400.a	DB2 Universal Database SQL CLI runtime
libdbm.a	New Database Manager (NDBM) interfaces
libdbx.a	dbx (debugger) utility support
libdl.a	Dynamic load runtime
libg.a	Debug support
libgaimisc.a	Internal X Windows support
libgair4.a	Internal X Windows support
libi18n.a	Internationalization runtime
libICE.a	Inter-Client Exchange library
libiconv.a	Character conversion runtime
libIM.a	Input method library
libl.a	lex support
libld.a	Object File Access Routine library
libm.a	IEEE Math library
libMrm.a	Motif Runtime library for UIL
libnsl.a	Transport Independent Remote Procedure Call (TI-RPC)
libpthdebug.a	Threads debug support
libpthreads.a	Threads runtime
libpthreads_compat.a	Old threads compatibility
libPW.a	Programmers Workbench library
librtl.a	Runtime linking runtime
libSM.a	X Session Management library
libtli.a	Transport Library Interface
libUil.a	Motif User Interface Language library
libxcurses.a	Curses library
libxti.a	X/Open Transport Library Interface
libX11.a	C interface for the X Window System protocol
libXaw.a	Athena Widget Set
libXext.a	Interfaces to X windows extensions
libXi.a	X Windows input processing

Library	Description
libxlf90_r.a	FORTRAN runtime
libxlfpthrds_compat.a	Old FORTRAN threads compatibility
libxlomp_ser.a	Open mp (multi-processing) support
libxlsmp.a	Symmetric mp (multiprocessing) support
libXm.a	Motif widget library
libXmu.a	Miscellaneous X Windows utility functions
libXtst.a	X Windows testing support
libXt.a	X Toolkit Intrinsics
liby.a	yacc support

Top | "i5/OS PASE APIs," on page 1 | APIs by category

i5/OS PASE Locales

i5/OS PASE includes a subset of the locales provided by AIX, supporting both 32-bit and 64-bit applications. i5/OS PASE locales are installed as symbolic links in directory /usr/lib/nls/loc.

The full name of any i5/OS PASE locale includes a code set name, which equates to the Coded Character Set Identifier (CCSID) shown in the table. Some locales also have a short name that exclude the code set part of the name. Any locale with a name ending in "@euro" uses the Euro as the currency symbol. Any locale with a name ending in "@preeuro" uses the traditional currency symbol.

Most i5/OS PASE locales are shipped with i5/OS language feature codes. Only locales in the base *CODE load and locales for installed language feature codes will exist on a particular i5/OS system.

	Locale Names				
Feature	**Short Name**	**Full Name**	**Language**	**Region**	**CCSID**
*CODE	be_BY	be_BY.ISO8859-5	Byelorussian	Byelorussian SSR	915
	BE_BY	BE_BY.UTF-8	Byelorussian	Byelorussian SSR	1208
	Et_EE	Et_EE.IBM-922	Estonian	Estonia	922
	ET_EE	ET_EE.UTF-8	Estonian	Estonia	1208
	UK_UA	UK_UA.UTF-8	Ukrainian	Ukraine	1208
	id_ID	id_ID.8859-15	Indonesian	Indonesia	923
	ID_ID	ID_ID.UTF-8	Indonesian	Indonesia	1208
	ms_MY	ms_MY.8859-15	Malay	Malaysia	923
MS_MY	MS_MY.UTF-8	Malay	Malaysia	1208	
2903	Lt_LT	Lt_LT.IBM-921	Lithuanian	Lithuania	921
	LT_LT	LT_LT.UTF-8	Lithuanian	Lithuania	1208
2904	Lv_LV	Lv_LV.IBM-921	Latvian	Latvia	921
	LV_LV	LV_LV.UTF-8	Latvian	Latvia	1208
2905	VI_VN	VI_VN.UTF-8	Vietnamese	Vietnam	1208

Feature	Locale Names		Language	Region	CCSID
	Short Name	Full Name			
2909	en_BE	en_BE.8859-15	English	Belgium	923
		en_BE.8859-15@preeuro	English	Belgium	923
	EN_BE	EN_BE.UTF-8	English	Belgium	1208
		EN_BE.UTF-8@preeuro	English	Belgium	1208
2911	sl_SI	sl_SI.ISO8859-2	Slovene	Slovenia	912
	SL_SI	SL_SI.UTF-8	Slovene	Slovenia	1208
2912	hr_HR	hr_HR.ISO8859-2	Croatian	Croatia	912
	HR_HR	HR_HR.UTF-8	Croatian	Croatia	1208
2913	mk_MK	mk_MK.ISO8859-5	Macedonian	Macedonia	915
	MK_MK	MK_MK.UTF-8	Macedonian	Macedonia	1208
2914	sh_SP	sh_SP.ISO8859-2	Serbian Latin	Yugoslavia	912
	SH_SP	SH_SP.UTF-8	Serbian Latin	Yugoslavia	1208
	sh_YU	sh_YU.ISO8859-2	Serbian Latin	Yugoslavia	912
	SH_YU	SH_YU.UTF-8	Serbian Latin	Yugoslavia	1208
	sr_SP	sr_SP.ISO8859-5	Serbian Cyrillic	Yugoslavia	915
	SR_SP	SR_SP.UTF-8	Serbian Cyrillic	Yugoslavia	1208
	sr_YU	sr_YU.ISO8859-5	Serbian Cyrillic	Yugoslavia	915
	SR_YU	SR_YU.UTF-8	Serbian Cyrillic	Yugoslavia	1208
2922	pt_PT	pt_PT.ISO8859-1	Portuguese	Portugal	819
		pt_PT.IBM-1252	Portuguese	Portugal	1252
		pt_PT.IBM-1252@preeuro	Portuguese	Portugal	1252
		pt_PT.8859-15	Portuguese	Portugal	923
		pt_PT.8859-15@preeuro	Portuguese	Portugal	923
	PT_PT	PT_PT.UTF-8	Portuguese	Portugal	1208
		PT_PT.UTF-8@preeuro	Portuguese	Portugal	1208
2923	nl_NL	nl_NL.ISO8859-1	Dutch	Netherlands	819
		nl_NL.IBM-1252	Dutch	Netherlands	1252
		nl_NL.IBM-1252@preeuro	Dutch	Netherlands	1252
		nl_NL.8859-15	Dutch	Netherlands	923
		nl_NL.8859-15@preeuro	Dutch	Netherlands	923
	NL_NL	NL_NL.UTF-8	Dutch	Netherlands	1208
		NL_NL.UTF-8@preeuro	Dutch	Netherlands	1208

	Locale Names				
Feature	Short Name	Full Name	Language	Region	CCSID
2924	en_AU	en_AU.8859-15	English	Australia	923
	EN_AU	EN_AU.UTF-8	English	Australia	1208
	en_CA	en_CA.8859-15	English	Canada	923
	EN_CA	EN_CA.UTF-8	English	Canada	1208
	en_GB	en_GB.ISO8859-1	English	Great Britain	819
		en_GB.IBM-1252	English	Great Britain	1252
		en_GB.IBM-1252@euro	English	Great Britain	1252
		en_GB.8859-15	English	Great Britain	923
		en_GB.8859-15@euro	English	Great Britain	923
	EN_GB	EN_GB.UTF-8	English	Great Britain	1208
	en_HK	en_HK.8859-15	English	Hong Kong	923
	EN_HK	EN_HK.UTF-8	English	Hong Kong	1208
	en_IE	en_IE.8859-15	English	Ireland	923
		en_IE.8859-15@preeuro	English	Ireland	923
	EN_IE	EN_IE.UTF-8	English	Ireland	1208
		EN_IE.UTF-8@preeuro	English	Ireland	1208
	en_IN	en_IN.8859-15	English	India	923
	EN_IN	EN_IN.UTF-8	English	India	1208
	en_NZ	en_NZ.8859-15	English	New Zealand	923
	EN_NZ	EN_NZ.UTF-8	English	New Zealand	1208
	en_PH	en_PH.8859-15	English	Philippines	923
	EN_PH	EN_PH.UTF-8	English	Philippines	1208
	en_SG	en_SG.8859-15	English	Singapore	923
	EN_SG	EN_SG.UTF-8	English	Singapore	1208
	en_US	en_US.ISO8859-1	English	United States	819
		en_US.8859-15	English	United States	923
	EN_US	EN_US.UTF-8	English	United States	1208
	en_ZA	en_ZA.8859-15	English	South Africa	923
	EN_ZA	EN_ZA.UTF-8	English	South Africa	1208
	» GU_IN	GU_IN.UTF-8	Gujarati	India	1208 «
	HI_IN	HI_IN.UTF-8	Hindi	India	1208
	» MR_IN	MR_IN.UTF-8	Marathi	India	1208 «
	» TA_IN	TA_IN.UTF-8	Tamil	India	1208 «
	» TE_IN	TE_IN.UTF-8	Telugu	India	1208 «

Feature	Locale Names		Language	Region	CCSID
	Short Name	Full Name			
2925	fi_FI	fi_FI.ISO8859-1	Finnish	Finland	819
		fi_FI.IBM-1252	Finnish	Finland	1252
		fi_FI.IBM-1252@preeuro	Finnish	Finland	1252
		fi_FI.8859-15	Finnish	Finland	923
		fi_FI.8859-15@preeuro	Finnish	Finland	923
	FI_FI	FI_FI.UTF-8	Finnish	Finland	1208
		FI_FI.UTF-8@preeuro	Finnish	Finland	1208
2926	da_DK	da_DK.ISO8859-1	Danish	Denmark	819
		da_DK.8859-15	Danish	Denmark	923
	DA_DK	DA_DK.UTF-8	Danish	Denmark	1208
2928	fr_FR	fr_FR.ISO8859-1	French	France	819
		fr_FR.IBM-1252	French	France	1252
		fr_FR.IBM-1252@preeuro	French	France	1252
		fr_FR.8859-15	French	France	923
		fr_FR.8859-15@preeuro	French	France	923
	FR_FR	FR_FR.UTF-8	French	France	1208
		FR_FR.UTF-8@preeuro	French	France	1208
2929	de_AT	de_AT.8859-15	German	Austria	923
		de_AT.8859-15@preeuro	German	Austria	923
	DE_AT	DE_AT.UTF-8	German	Austria	1208
		DE_AT.UTF-8@preeuro	German	Austria	1208
	de_DE	de_DE.ISO8859-1	German	Germany	819
		de_DE.IBM-1252	German	Germany	1252
		de_DE.IBM-1252@preeuro	German	Germany	1252
		de_DE.8859-15	German	Germany	923
		de_DE.8859-15@preeuro	German	Germany	923
	DE_DE	DE_DE.UTF-8	German	Germany	1208
		DE_DE.UTF-8@preeuro	German	Germany	1208

Feature	Locale Names		Language	Region	CCSID
	Short Name	Full Name			
2931 (part 1)	ca_ES	ca_ES.ISO8859-1	Catalan	Spain	819
		ca_ES.IBM-1252	Catalan	Spain	1252
		ca_ES.IBM-1252@preeuro	Catalan	Spain	1252
		ca_ES.8859-15	Catalan	Spain	923
		ca_ES.8859-15@preeuro	Catalan	Spain	923
	CA_ES	CA_ES.UTF-8	Catalan	Spain	1208
		CA_ES.UTF-8@preeuro	Catalan	Spain	1208
	es_AR	es_AR.8859-15	Spanish	Argentina	923
	ES_AR	ES_AR.UTF-8	Spanish	Argentina	1208
	es_BO	es_BO.8859-15	Spanish	Bolivia	923
	ES_BO	ES_BO.UTF-8	Spanish	Bolivia	1208
	es_CL	es_CL.8859-15	Spanish	Chile	923
	ES_CL	ES_CL.UTF-8	Spanish	Chile	1208
	es_CO	es_CO.8859-15	Spanish	Columbia	923
	ES_CO	ES_CO.UTF-8	Spanish	Columbia	1208
	es_CR	es_CR.8859-15	Spanish	Costa Rica	923
	ES_CR	ES_CR.UTF-8	Spanish	Costa Rica	1208
	es_DO	es_DO.8859-15	Spanish	Dominican Republic	923
	ES_DO	ES_DO.UTF-8	Spanish	Dominican Republic	1208
	es_EC	es_EC.8859-15	Spanish	Ecuador	923
	ES_EC	ES_EC.UTF-8	Spanish	Ecuador	1208
	es_ES	es_ES.ISO8859-1	Spanish	Spain	819
		es_ES.IBM-1252	Spanish	Spain	1252
		es_ES.IBM-1252@preeuro	Spanish	Spain	1252
		es_ES.8859-15	Spanish	Spain	923
		es_ES.8859-15@preeuro	Spanish	Spain	923

Feature	Locale Names		Language	Region	CCSID
	Short Name	Full Name			
2931 (part 2)	ES_ES	ES_ES.UTF-8	Spanish	Spain	1208
		ES_ES.UTF-8@preeuro	Spanish	Spain	1208
	es_GT	es_GT.8859-15	Spanish	Guatemala	923
	ES_GT	ES_GT.UTF-8	Spanish	Guatemala	1208
	es_HN	es_HN.8859-15	Spanish	Honduras	923
	ES_HN	ES_HN.UTF-8	Spanish	Honduras	1208
	es_MX	es_MX.8859-15	Spanish	Mexico	923
	ES_MX	ES_MX.UTF-8	Spanish	Mexico	1208
	es_NI	es_NI.8859-15	Spanish	Nicaragua	923
	ES_NI	ES_NI.UTF-8	Spanish	Nicaragua	1208
	es_PA	es_PA.8859-15	Spanish	Panama	923
	ES_PA	ES_PA.UTF-8	Spanish	Panama	1208
	es_PE	es_PE.8859-15	Spanish	Peru	923
	ES_PE	ES_PE.UTF-8	Spanish	Peru	1208
	es_PR	es_PR.8859-15	Spanish	Puerto Rico	923
	ES_PR	ES_PR.UTF-8	Spanish	Puerto Rico	1208
	es_PY	es_PY.8859-15	Spanish	Paraguay	923
	ES_PY	ES_PY.UTF-8	Spanish	Paraguay	1208
	es_SV	es_SV.8859-15	Spanish	El Salvador	923
	ES_SV	ES_SV.UTF-8	Spanish	El Salvador	1208
	es_US	es_US.8859-15	Spanish	United States	923
	ES_US	ES_US.UTF-8	Spanish	United States	1208
	es_UY	es_UY.8859-15	Spanish	Uruguay	923
	ES_UY	ES_UY.UTF-8	Spanish	Uruguay	1208
	es_VE	es_VE.8859-15	Spanish	Venezuela	923
	ES_VE	ES_VE.UTF-8	Spanish	Venezuela	1208
2932	it_IT	it_IT.ISO8859-1	Italian	Italy	819
		it_IT.IBM-1252	Italian	Italy	1252
		it_IT.IBM-1252@preeuro	Italian	Italy	1252
		it_IT.8859-15	Italian	Italy	923
		it_IT.8859-15@preeuro	Italian	Italy	923
	IT_IT	IT_IT.UTF-8	Italian	Italy	1208
		IT_IT.UTF-8@preeuro	Italian	Italy	1208
2933	no_NO	no_NO.ISO8859-1	Norwegian	Norway	819
		no_NO.8859-15	Norwegian	Norway	923
	NO_NO	NO_NO.UTF-8	Norwegian	Norway	1208
2937	sv_SE	sv_SE.ISO8859-1	Swedish	Sweden	819
		sv_SE.8859-15	Swedish	Sweden	923
	SV_SE	SV_SE.UTF-8	Swedish	Sweden	1208

Feature	Locale Names		Language	Region	CCSID
	Short Name	Full Name			
2939	de_CH	de_CH.ISO8859-1	German	Switzerland	819
		de_CH.8859-15	German	Switzerland	923
	DE_CH	DE_CH.UTF-8	German	Switzerland	1208
	de_LU	de_LU.8859-15	German	Luxembourg	923
		de_LU.8859-15@preeuro	German	Luxembourg	923
	DE_LU	DE_LU.UTF-8	German	Luxembourg	1208
		DE_LU.UTF-8@preeuro	German	Luxembourg	1208
2940	fr_CH	fr_CH.ISO8859-1	French	Switzerland	819
		fr_CH.8859-15	French	Switzerland	923
	FR_CH	FR_CH.UTF-8	French	Switzerland	1208
2942	it_CH	it_CH.8859-15	Italian	Switzerland	923
	IT_CH	IT_CH.UTF-8	Italian	Switzerland	1208

Feature	Locale Names		Language	Region	CCSID
	Short Name	Full Name			
2954	ar_AA	ar_AA.ISO8859-6	Arabic	Arabic Countries	1089
	» Ar_AA	Ar_AA.IBM-1046	Arabic	Arabic Countries	1046 «
	AR_AA	AR_AA.UTF-8	Arabic	Arabic Countries	1208
	ar_AE	ar_AE.ISO8859-6	Arabic	United Arab Emirates	1089
	AR_AE	AR_AE.UTF-8	Arabic	United Arab Emirates	1208
	ar_BH	ar_BH.ISO8859-6	Arabic	Bahrain	1089
	AR_BH	AR_BH.UTF-8	Arabic	Bahrain	1208
	ar_DZ	ar_DZ.ISO8859-6	Arabic	Algeria	1089
	AR_DZ	AR_DZ.UTF-8	Arabic	Algeria	1208
	ar_EG	ar_EG.ISO8859-6	Arabic	Egypt	1089
	AR_EG	AR_EG.UTF-8	Arabic	Egypt	1208
	ar_JO	ar_JO.ISO8859-6	Arabic	Jordan	1089
	AR_JO	AR_JO.UTF-8	Arabic	Jordan	1208
	ar_KW	ar_KW.ISO8859-6	Arabic	Kuwait	1089
	AR_KW	AR_KW.UTF-8	Arabic	Kuwait	1208
	ar_LB	ar_LB.ISO8859-6	Arabic	Lebanon	1089
	AR_LB	AR_LB.UTF-8	Arabic	Lebanon	1208
	ar_MA	ar_MA.ISO8859-6	Arabic	Morocco	1089
	AR_MA	AR_MA.UTF-8	Arabic	Morocco	1208
	ar_OM	ar_OM.ISO8859-6	Arabic	Oman	1089
	AR_OM	AR_OM.UTF-8	Arabic	Oman	1208
	ar_QA	ar_QA.ISO8859-6	Arabic	Qatar	1089
	AR_QA	AR_QA.UTF-8	Arabic	Qatar	1208
	ar_SA	ar_SA.ISO8859-6	Arabic	Saudi Arabia	1089
	AR_SA	AR_SA.UTF-8	Arabic	Saudi Arabia	1208
	ar_SY	ar_SY.ISO8859-6	Arabic	Syrian Arab Republic	1089
	AR_SY	AR_SY.UTF-8	Arabic	Syrian Arab Republic	1208
	ar_TN	ar_TN.ISO8859-6	Arabic	Tunisia	1089
	AR_TN	AR_TN.UTF-8	Arabic	Tunisia	1208
	ar_YE	ar_YE.ISO8859-6	Arabic	Yemen	1089
	AR_YE	AR_YE.UTF-8	Arabic	Yemen	1208
2956	tr_TR	tr_TR.ISO8859-9	Turkish	Turkey	920
	TR_TR	TR_TR.UTF-8	Turkish	Turkey	1208
2957	el_GR	el_GR.ISO8859-7	Greek	Greece	813
	EL_GR	EL_GR.UTF-8	Greek	Greece	1208
2958	is_IS	is_IS.ISO8859-1	Icelandic	Iceland	819
		is_IS.8859-15	Icelandic	Iceland	923
	IS_IS	IS_IS.UTF-8	Icelandic	Iceland	1208
2961	iw_IL	iw_IL.ISO8859-8	Hebrew	Israel	916
	HE_IL	HE_IL.UTF-8	Hebrew	Israel	1208

	Locale Names				
Feature	**Short Name**	**Full Name**	**Language**	**Region**	**CCSID**
2962	ja_JP	ja_JP.IBM-eucJP	Japanese	Japan	33722
	Ja_JP	ja_JP.IBM-943	Japanese	Japan	943
	JA_JP	JA_JP.UTF-8	Japanese	Japan	1208
2963	nl_BE	nl_BE.ISO8859-1	Dutch	Belgium	819
		nl_BE.IBM-1252	Dutch	Belgium	1252
		nl_BE.IBM-1252@preeuro	Dutch	Belgium	1252
		nl_BE.8859-15	Dutch	Belgium	923
		nl_BE.8859-15@preeuro	Dutch	Belgium	923
	NL_BE	NL_BE.UTF-8	Dutch	Belgium	1208
		NL_BE.UTF-8@preeuro	Dutch	Belgium	1208
2966	fr_BE	fr_BE.ISO8859-1	French	Belgium	819
		fr_BE.IBM-1252	French	Belgium	1252
		fr_BE.IBM-1252@preeuro	French	Belgium	1252
		fr_BE.8859-15	French	Belgium	923
		fr_BE.8859-15@preeuro	French	Belgium	923
	FR_BE	FR_BE.UTF-8	French	Belgium	1208
		FR_BE.UTF-8@preeuro	French	Belgium	1208
	fr_LU	fr_LU.8859-15	French	Luxembourg	923
		fr_LU.8859-15@preeuro	French	Luxembourg	923
	FR_LU	FR_LU.UTF-8	French	Luxembourg	1208
		FR_LU.UTF-8@preeuro	French	Luxembourg	1208
2972	th_TH	TH_TH.TIS-620	Thai	Thailand	874
	TH_TH	TH_TH.UTF-8	Thai	Thailand	1208
2974	bg_BG	bg_BG.ISO8859-5	Bulgarian	Bulgaria	915
	BG_BG	BG_BG.UTF-8	Bulgarian	Bulgaria	1208
2975	cs_CZ	cs_CZ.ISO8859-2	Czech	Czech Republic	912
	CS_CZ	CS_CZ.UTF-8	Czech	Czech Republic	1208
2976	hu_HU	hu_HU.ISO8859-2	Hungarian	Hungary	912
	HU_HU	HU_HU.UTF-8	Hungarian	Hungary	1208
2978	pl_PL	pl_PL.ISO8859-2	Polish	Poland	912
	PL_PL	PL_PL.UTF-8	Polish	Poland	1208
2979	» KK_KZ	KK_KZ.UTF-8	Kazakh	Kazakhstan	1208 «
	ru_RU	ru_RU.ISO8859-5	Russian	Russia	915
	RU_RU	RU_RU.UTF-8	Russian	Russia	1208
2980	pt_BR	pt_BR.ISO8859-1	Portuguese	Brazil	819
		pt_BR.8859-15	Portuguese	Brazil	923
	PT_BR	PT_BR.UTF-8	Portuguese	Brazil	1208
2981	fr_CA	fr_CA.ISO8859-1	French	Canada	819
		fr_CA.8859-15	French	Canada	923
	FR_CA	FR_CA.UTF-8	French	Canada	1208

Feature	Locale Names		Language	Region	CCSID
	Short Name	Full Name			
2986	ko_KR	ko_KR.IBM-eucKR	Korean	Korea	970
	KO_KR	KO_KR.UTF-8	Korean	Korea	1208
2987	zh_TW	zh_TW.IBM-eucTW	Traditional Chinese	Taiwan	964
	Zh_TW	ZH_TW.big5	Traditional Chinese	Taiwan	950
	zh_TW	ZH_TW.UTF-8	Traditional Chinese	Taiwan	1208
2989	zh_CN	zh_CN.IBM-eucCN	Simplified Chinese	People's Republic of China	1383
	Zh_CN	zh_CN.GB18030	Simplified Chinese	People's Republic of China	1386
	ZH_CN	ZH_CN.UTF-8	Simplified Chinese	People's Republic of China	1208
	ZH_HK	ZH_HK.UTF-8	Simplified Chinese	Hong Kong	1208
	ZH_SG	ZH_SG.UTF-8	Simplified Chinese	Singapore	1208
2992	ro_RO	ro_RO.ISO8859-2	Romanian	Romania	912
	RO_RO	RO_RO.UTF-8	Romanian	Romania	1208
2994	sk_SK	sk_SK.ISO8859-2	Slovak	Slovakia	912
	SK_SK	SK_SK.UTF-8	Slovak	Slovakia	1208
2995	sq_AL	sq_AL.ISO8859-1	Serbian Cyrillic	Yugoslavia	915
		sq_AL.8859-15	Serbian Cyrillic	Yugoslavia	923
	SQ_AL	SQ_AL.UTF-8	Serbian Cyrillic	Yugoslavia	1208

Top | "i5/OS PASE APIs," on page 1 | APIs by category

i5/OS PASE Environment Variables

Overview

i5/OS PASE environment variables are independent of ILE environment variables. Setting a variable in one environment has no effect on the other environment, but several system interfaces allow you to copy variables between environments:

- The **Qp2RunPase** API lets you specify any list of environment variables you want to initialize for the i5/OS PASE program. See "Qp2RunPase()—Run an i5/OS PASE Program" on page 20 (Qp2RunPase) documentation for more information.
- The **QP2SHELL** and **QP2TERM** APIs initialize the i5/OS PASE environment with a copy of nearly all ILE environment variables. See "QP2SHELL() and QP2SHELL2()—Run an i5/OS PASE Shell Program" on page 2 (QP2SHELL) documentation for more information.
- The **systemCL** i5/OS PASE runtime function copies nearly all i5/OS PASE environment variables to the ILE environment for option **SYSTEMCL_ENVIRON**. See "systemCL()—Run a CL Command for i5/OS PASE" on page 42 (systemCL) documentation for more information.
- The i5/OS PASE **system** utility copies nearly all i5/OS PASE environment variables to the ILE environment for option **-e**. See Run a CL Command (i5/OS PASE **system** utility) documentation for more information.

Special i5/OS PASE Environment Variables

Some i5/OS PASE runtime behaviors are different from AIX because of differences between the two operating systems. You can use these i5/OS PASE environment variables to control some of the differences:

PASE_EXEC_QOPENSYS

PASE_EXEC_QOPENSYS can be used to prevent the system from searching the /QOpenSys file system for an absolute path (starting with ″/″) specified as an argument to **exec** or **Qp2RunPase**, or in the first line of a shell script. The system normally searches the /QOpenSys file system if the absolute path name for an i5/OS PASE program or script cannot be opened or is not a regular bytestream file. i5/OS directory /usr/bin contains links to QShell utilities that cannot run as i5/OS PASE programs, so searching /QOpenSys allows more AIX programs and shell scripts to run unchanged (using i5/OS PASE utilities in directory /QOpenSys/usr/bin). The system does not do an extended search in the /QOpenSys file system if the i5/OS PASE shell or other program that calls **exec** or **Qp2RunPase** has changed credentials (**setuid** or **setgid**) or if the i5/OS PASE environment specifies **PASE_EXEC_QOPENSYS=N**.

PASE_FORK_JOBNAME

PASE_FORK_JOBNAME specifies the i5/OS job name for a new process created with the **fork()** or **f_fork()** function. Only the first 10 characters in the string are used, and lowercase characters are converted to uppercase. The specified value is ignored and a default job name is used if the string does not follow i5/OS simple name rules (first character alphabetic and subsequent characters alphameric or underscore). Prestarted job are never used when a fork job name is specified. See “fork400()and f_fork400()—Create A New Process with i5/OS PASE Options” on page 28 (fork400 or f_fork400) for information about specifying the i5/OS job name for specific fork operations.

Some i5/OS PASE shells (including the default Korn shell) do not set environment variables for exported variables in the shell process itself. Setting **PASE_FORK_JOBNAME** in such a shell does not control job names for first-order utility processes created by that shell, but can control job names for processes forked by a utility started by the shell.

PASE_MAXDATA64

PASE_MAXDATA64 specifies the maximum number of 256MB segments provided for brk (heap) storage in a 64-bit i5/OS PASE program. If **PASE_MAXDATA64** is omitted or contains an invalid value (either non-numeric or less than one), a default of 256 segments (64GB) is used. **PASE_MAXDATA64** has no effect on 32-bit i5/OS PASE programs, and it must be set either in the initial environment passed to **Qp2RunPase** or before running **exec** for a 64-bit i5/OS PASE program.

PASE_MAXSHR64

PASE_MAXSHR64 specifies the maximum number of 256MB segments provided for shared memory (shmat and mmap) in a 64-bit i5/OS PASE program. If **PASE_MAXSHR64** is omitted or contains an invalid value (either non-numeric or less than one), a default of 256 segments (64GB) is used. **PASE_MAXSHR64** has no effect on 32-bit i5/OS PASE programs, and it must be set either in the initial environment passed to **Qp2RunPase** or before running **exec** for a 64-bit i5/OS PASE program.

PASE_STDIO_ISATTY

The default behavior of the i5/OS PASE **isatty** runtime function returns true for file descriptors 0, 1, and 2 (stdin, stdout, and stderr), regardless of whether the open file is a tty device. Setting i5/OS PASE environment variable **PASE_STDIO_ISATTY** to N, either in the initial environment passed to **Qp2RunPase** or before the first invocation of **isatty**, causes **isatty** to return an accurate indication of whether the open file is a tty device.

PASE_SYSCALL_NOSIGILL

The i5/OS PASE kernel exports some system calls that are implemented by the AIX kernel but are unsupported by i5/OS PASE. The default behavior for any unsupported syscall is to send exception message MCH3204, which the system converts to i5/OS PASE signal **SIGILL**. The unsupported syscall returns a function result of -1 with i5/OS-unique errno EUNKNOWN (3474) if the signal is ignored or the handler returns. Message MCH3204 appears in the i5/OS job log to provide the name of the unsupported system call and the i5/OS PASE instruction address that caused the error. The message may also include the internal dump identifier for a VLOG entry that contains this information:

```
syscall number (GPR2 value)
i5/OS PASE instruction address
Link register value
GPR3-10 values (if available, or zero otherwise)
syscall name (if known, converted to uppercase)
```

i5/OS PASE programs can suppress the exception message and **SIGILL** signal for unsupported system calls by setting environment variable **PASE_SYSCALL_NOSIGILL** either in the initial environment passed to **Qp2RunPase** or before running **exec**. **PASE_SYSCALL_NOSIGILL** is ignored if the i5/OS PASE program has the S_ISUID or S_ISGID attribute, but otherwise is interpreted as a list of syscall function names with optional errno values, delimited by colons. The colon-delimited values must take one of these forms:

```
syscall_name
syscall_name=errno_name     (errno_name is EINVAL, EPERM, and so on)
syscall_name=errno_number   (errno_number is 0-127)
```

SIGILL is suppressed for any *syscall_name* in the list that is recognized as an i5/OS PASE system call. The first or only entry in the list may use a special *syscall_name* of "ALL" to set a default behavior for all unsupported syscalls. Any entry in the list that is not an i5/OS PASE syscall name is ignored, and specifying the name of a syscall that is supported by the i5/OS PASE kernel has no effect on the operation of that syscall.

Any syscall in the **PASE_SYSCALL_NOSIGILL** list that is unsupported by the i5/OS PASE kernel returns a function result of -1 with the specified errno value (defaulting to ENOSYS) except that specifying errno_number of 0 causes the unsupported syscall to return a function result of zero (without setting errno). An invalid errno_name or errno_number defaults to ENOSYS.

For example, the following **PASE_SYSCALL_NOSIGILL** value suppresses **SIGILL** for all unsupported syscalls. "quotactl" returns EPERM and "audit" returns function result of zero, while all other unsupported syscalls return ENOSYS:

```
export PASE_SYSCALL_NOSIGILL=ALL:quotactl=EPERM:audit=0
```

Note: **PASE_SYSCALL_NOSIGILL** is not intended for production programs. It is provided as a convenience for feasibility testing using unchanged AIX binaries that need to be modified for production.

PASE_THREAD_ATTACH
If i5/OS PASE environment variable **PASE_THREAD_ATTACH** is set to Y when an i5/OS PASE program runs libpthreads.a initialization (usually at program startup), an ILE thread that was not started by i5/OS PASE will be attached to i5/OS PASE when it calls an i5/OS PASE procedure (using **Qp2CallPase** or **Qp2CallPase2**). Once an ILE thread has attached to i5/OS PASE, that thread is subject to asynchronous interruption for i5/OS PASE functions such as signal handling and thread cancellation. In particular, the thread will be canceled as part of ending the i5/OS PASE program (when **exit** runs or i5/OS PASE processing terminates for a signal).

PASE_UNLIMITED_PATH_MAX
The i5/OS Integrated File System supports longer path names than the value of **PATH_MAX** (1023) in AIX header file <**limits.h**>. Setting i5/OS PASE environment variable **PASE_UNLIMITED_PATH_MAX** to Y, either in the initial environment passed to **Qp2RunPase**

or before running **exec**, allows an i5/OS PASE program to access objects with long path names. i5/OS PASE loader functions and some library runtime functions can fail with path names longer than AIX **PATH_MAX**.

PASE_USRGRP_LOWERCASE
i5/OS user names and group names are case-insensitive, but the system stores and returns them in uppercase. i5/OS PASE runtime functions that return user names and group names (getpwnam, getpwuid, getgrnam, and getgrgid) default to converting them to lowercase unless i5/OS PASE environment variable **PASE_USRGRP_LOWERCASE** is set to N.

Top | "i5/OS PASE APIs," on page 1 | APIs by category

i5/OS PASE Signal Handling

i5/OS PASE Signals and ILE Signals

OS/PASE signals and POSIX/ILE signals are independent, so it is not possible to directly call a handler for one signal type by raising the other type of signal. However, the "Qp2SignalPase()—Post an i5/OS PASE Signal" on page 24 (Qp2SignalPase) API can be used as the handler for any ILE signal to post a corresponding i5/OS PASE signal. An i5/OS PASE program can also define handlers for i5/OS PASE signals that call ILE procedures to post equivalent ILE signals. Program **QP2SHELL** and the i5/OS PASE **fork** function always setup handlers to map every ILE signal to a corresponding i5/OS PASE signal.

i5/OS Messages and i5/OS PASE Programs

Many i5/OS applications and system functions report errors with exception messages sent to program call message queues. See Message Handling Terms and Concepts for information about exception messages and program call message queues.

The system only creates program call message queues for ILE procedures and OMI programs. Any machine exception caused by an operation inside an i5/OS PASE program (such as MCH0601 for a storage reference error) is sent to the program call message queue for an (internal) ILE procedure in service program QP2USER. This ILE procedure is also the apparent caller of any ILE procedure the i5/OS PASE program calls directly (using _ILECALLX or _ILECALL), so any i5/OS message the called procedure sends to its caller goes to the same message queue used for machine exceptions.

i5/OS Exceptions and i5/OS PASE Signals

The ILE procedure in service program QP2USER that runs i5/OS PASE programs handles any exception and converts it to an i5/OS PASE signal, the same way POSIX/ILE C runtime converts exceptions to ILE signals. The specific signal used depends on the i5/OS message identifier for the exception. i5/OS PASE and ILE use different signal numbers, but both map any specific message identifier to the same signal name (such as SIGSEGV). See the WebSphere Development Studio: ILE C/C++ Programmer's Guide for details.

An i5/OS PASE signal handler can determine whether a signal is associated with an exception message by inspecting field *msgkey* in the ucontext_t_os400 structure (declared in header file as400_types.h) that is passed as an argument to the handler. A non-zero value is the message reference key for the i5/OS message that caused the signal. Zero indicates the signal is not associated with an i5/OS message (which is always true for asynchronous signals). The i5/OS PASE program can use the message reference key to receive the exception message (see "QMHRCVPM()—Receive Program Message for i5/OS PASE" on page 33) for more details about the error.

Top | "i5/OS PASE APIs," on page 1 | 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:

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

Terms and Conditions

Printed in USA