libdeci4p Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

libdeci4p Processing

sceKernelDeci4pOpen

Open DECI4p protocol socket

Definition

```
#include <libdeci4p.h>
SceUID sceKernelDeci4pOpen(
        const char *protoname,
        SceUInt32 protonum,
        SceSize bufsize
);
```

Arguments

protoname	Name of protocol driver Whether the name is unique is not checked. The name can be up to 31 bytes long.
protonum	Protocol number
bufsize	Packet buffer size (unit: byte) Specify the size from 16 bytes to 1 MB in multiples of 4.

Return Values

Upon normal termination, becomes a positive value and returns the DECI4p protocol socket ID.

Upon occurrence of an error, becomes a negative value and returns one of the following error codes.

Value	(Number)	Description
`SCE_KERNEL_ERROR_DECI4P_UNKNOWN`	0x80080800	Undefined error not listed below
`SCE_KERNEL_ERROR_DECI4P_ALREADYUSE_PROTOCOL`	0x80080801	Protocol number that is already used
`SCE_KERNEL_ERROR_DECI4P_ILLEGAL_SID`	0x80080802	Invalid socket ID
`SCE_KERNEL_ERROR_DECI4P_ILLEGAL_ADDR`	0x80080804	Invalid address
`SCE_KERNEL_ERROR_DECI4P_ILLEGAL_SIZE`	0x80080805	Invalid size
`SCE_KERNEL_ERROR_DECI4P_ILLEGAL_PROTOCOL`	0x80080806	Invalid protocol number
`SCE_KERNEL_ERROR_DECI4P_NO_MEMORY`	0x80080807	Insufficient memory
`SCE_KERNEL_ERROR_DECI4P_TOOMANY_PROTOCOL`	0x80080809	Too many protocols

Description

Specify the DECI4p protocol number to be used and register the protocol driver. The socket ID returned here is used from then for function calls.

The buffer used by the protocols for communication is allocated from the dedicated memory partition of the Development Kit. DECI4p packets of a size that exceeds the buffer size cannot be received.

sceKernelDeci4pClose

Close DECI4p protocol socket

Definition

```
#include <libdeci4p.h>
SceInt32 sceKernelDeci4pClose(
        SceUID socketid
);
```

Arguments

socketid	DECI4p protocol socket ID

Return Values

Upon normal termination, returns `SCE_OK` (=0).

Returns the following error code upon an error.

Value	(Number)	Description
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_SID	0x80080802	Invalid socket ID

Description

This function closes the open DECI4p socket.

See Also

`sceKernelDeci4pOpen()`

sceKernelDeci4pRead

Receive data from DECI4p protocol socket

Definition

```
#include <libdeci4p.h>
SceInt32 sceKernelDeci4pRead(
        SceUID socketid,
        void *buffer,
        SceSize size,
        SceUInt32 reserved
);
```

Arguments

socketid	DECI4p protocol socket ID
buffer	Address of DECI4p payload data receive buffer
size	Size of DECI4p payload data receive buffer
reserved	Specify 0

Return Values

Upon normal termination, returns a receive size of 0 or larger.

Returns one of the following error codes upon an error.

Value	(Number)	Description
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_SID	0x80080802	Invalid socket ID
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_ADDR	0x80080804	Invalid address
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_SIZE	0x80080805	Invalid size
SCE_KERNEL_ERROR_DECI4P_NO_MEMORY	0x80080807	Insufficient memory
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_RESERVED	0x80080808	Invalid value
SCE_KERNEL_ERROR_DECI4P_NO_CONNECT	0x8008080A	Not connected to development host computer
SCE_KERNEL_ERROR_DECI4P_NO_PROTO	0x8008080B	Protocol not registered in Target Manager API
SCE_KERNEL_ERROR_DECI4P_TOOSMALL_BUFFER	0x8008080C	Buffer is too small

Description

This function receives data from an open DECI4p socket.

Reception is not possible when a buffer size smaller than the received DECI4p packet size is specified.

See Also

`sceKernelDeci4pOpen()`

sceKernelDeci4pWrite

Send data to DECI4p protocol socket

Definition

```
#include <libdeci4p.h>
SceInt32 sceKernelDeci4pWrite(
        SceUID socketid,
        void *buffer,
        SceSize size,
        SceUInt32 reserved
);
```

Arguments

socketid	DECI4p protocol socket ID
buffer	Address of DECI4p payload data send buffer
size	DECI4p payload data send size
reserved	Specify 0.

Return Values

Upon normal termination, returns a send size of 0 or larger.

Returns one of the following error codes upon an error.

Value	(Number)	Description
`SCE_KERNEL_ERROR_DECI4P_UNKNOWN`	0x80080800	Undefined error not listed below
`SCE_KERNEL_ERROR_DECI4P_ILLEGAL_SID`	0x80080802	Invalid socket ID
`SCE_KERNEL_ERROR_DECI4P_ILLEGAL_ADDR`	0x80080804	Invalid address
`SCE_KERNEL_ERROR_DECI4P_ILLEGAL_SIZE`	0x80080805	Invalid size
`SCE_KERNEL_ERROR_DECI4P_ILLEGAL_PROTOCOL`	0x80080806	Invalid protocol number
`SCE_KERNEL_ERROR_DECI4P_NO_MEMORY`	0x80080807	Insufficient memory
`SCE_KERNEL_ERROR_DECI4P_ILLEGAL_RESERVED`	0x80080808	Invalid value
`SCE_KERNEL_ERROR_DECI4P_NO_CONNECT`	0x8008080A	Not connected to development host computer
`SCE_KERNEL_ERROR_DECI4P_NO_PROTO`	0x8008080B	Protocol not registered in Target Manager API

Description

This function sends data to an open DECI4p socket.

See Also

`sceKernelDeci4pOpen()`

SceKernelDeci4pCallback

DECI4p socket callback function prototype

Definition

```
#include <libdeci4p.h>
typedef SceInt32 ( *SceKernelDeci4pCallback )(
        SceUID notifyId,
        SceInt32 notifyCount,
        SceInt32 callbackArg,
        void *pCommon
);
```

Arguments

notifyId	`SCE_UID_INVALID_UID`
notifyCount	Number of notifications
callbackArg	Notified events
pCommon	Callback parameter of application definition registered with `sceKernelCreateCallback()`

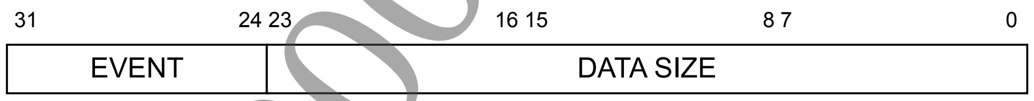
Return Values

Always returns `SCE_OK` (=0).

Description

This is a prototype of the callback function for receiving DECI4p socket event notifications.

To receive notification callback from a DECI4p socket, register the `SceUID` value of the callback generated with the `sceKernelCreateCallback()` function using the `sceKernelDeci4pRegisterCallback()` function. The notified event *`callbackArg`* is defined as follows.



Bits 31 to 24 are defined as events through logical OR expression.

EVENT	Description
`SCE_KERNEL_DECI4P_CALLBACKARG_DATA_READY`	Data received
`SCE_KERNEL_DECI4P_CALLBACKARG_NOCONNECT`	Not connected to development host computer
`SCE_KERNEL_DECI4P_CALLBACKARG_NOPROTOCOL`	No protocol is registered

Bits 23 to 0 define the size during data reception.

See Also

`sceKernelDeci4pRegisterCallback()`,`sceKernelCreateCallback()`

sceKernelDeci4pRegisterCallback

Register DECI4p socket callback function

Definition

```
#include <libdeci4p.h>
int32_t sceKernelDeci4pRegisterCallback(
        SceUID socketid,
        SceUID cbid
);
```

Arguments

socketid	DECI4p protocol socket ID
cbid	Callback ID

Return Values

Upon normal termination, returns `SCE_OK` (=0).

Returns one of the following error codes upon an error.

Value	(Number)	Description
`SCE_KERNEL_ERROR_DECI4P_ILLEGAL_SID`	0x80080802	Invalid socket ID
`SCE_KERNEL_ERROR_DECI4P_ILLEGAL_CBID`	0x80080803	Invalid callback ID

Description

This function registers callback function for receiving DECI4p socket event notifications. Callbacks are registered by socket. Callbacks that are registered with this function cannot be deregistered.

After closing a socket with the `sceKernelDeci4pClose()` function, delete callbacks with the `sceKernelDeleteCallback()` function.

See Also

`sceKernelCreateCallback(), sceKernelDeleteCallback()`

sceKernelDeci4pIsProcessAttached

Get host debugger connection status

Definition

```
#include <libdeci4p.h>
SceInt32 sceKernelDeci4pIsProcessAttached(
        void
);
```

Arguments

None

Return Values

Value	Description
0	Host debugger is disconnected
Value other than 0	Host debugger is connected

Description

This function enables the program that called this function to check whether or not a connection is established to it from the host debugger on the development host computer.

sceKernelDeci4pEnableWatchpoint

Enable data breakpoint (watchpoint)

Definition

```
#include <libdeci4p.h>
SceInt32 sceKernelDeci4pEnableWatchpoint(
        void
);
```

Arguments

None

Return Values

Value	Description
Negative value	Error code

Description

This function enables a data breakpoint (watchpoint) that was disabled with the `sceKernelDeci4pDisableWatchpoint()` function.

sceKernelDeci4pDisableWatchpoint

Disable data breakpoint (watchpoint)

Definition

```
#include <libdeci4p.h>
SceInt32 sceKernelDeci4pDisableWatchpoint(
        void
);
```

Arguments

None

Return Values

Value	Description
Negative value	Error code

Description

This function disables a data breakpoint (watchpoint) set with the host debugger. It can be re-enabled with the `sceKernelDeci4pEnableWatchpoint()` function.

SceKernelDeci4pCreateHostProcessParam

Startup parameters for application in development host computer

Definition

```
#include <libdeci4p.h>
#define SCE_KERNEL_DECI4P_HOST_PROCESS_MAX_STR_SIZE (1024)
typedef struct SceKernelDeci4pCreateHostProcessParam {
        SceUInt32 flags;
        SceUInt32 reserved0;
        const char *pathName;
        const char *cmdLine;
        const char *workDir;
        SceUInt32 reserved1[8];
} SceKernelDeci4pCreateHostProcessParam;
```

Members

flags	Startup options
reserved0	Reserved area (specify 0)
pathName	Path name of executable file in development host computer
cmdLine	Command line argument to pass to application, or NULL
workDir	Current directory upon application execution, or NULL (refer to Description)
reserved1	Reserved area (fill with all 0s)

Description

This structure represents the parameters for starting an application in a development host computer.

It is used when starting an application with `sceKernelDeci4pCreateHostProcess()` or `sceKernelDeci4pCreateHostProcessAndWait()`.

For *flags*, specify the settings related to application window display and the settings related to the current directory upon application execution. Specify the bitwise OR of each value.

One of the following values can be specified for the application window settings.

Value	Description
SCE_KERNEL_DECI4P_HOST_PROCESS_WINDOW_SHOW	Show window
SCE_KERNEL_DECI4P_HOST_PROCESS_WINDOW_HIDE	Hide window

One of the following values can be set as the current directory upon application execution.

Value	Description
SCE_KERNEL_DECI4P_HOST_PROCESS_WORKDIR_FSROOT	Development host computer file server directory (specify NULL for *workDir*)
SCE_KERNEL_DECI4P_HOST_PROCESS_WORKDIR_EXECUTABLE	Parent directory of executable file specified with *pathName* (specify NULL for *workDir*)
SCE_KERNEL_DECI4P_HOST_PROCESS_WORKDIR_OVERRIDE	Directory specified with *workDir*

For *pathName*, specify the path name of the executable file of the application to start beginning with host0:. Both relative paths from the file server directory and absolute paths can be used.

In addition, environment variables such as %WINDIR% can be used for *pathName*.

For *cmdLine*, it is possible to specify a command line argument to pass to the application. If an argument is not required, specify NULL.

For *`workDir`*, specify the path name of the current directory upon application execution. For *`workDir`*, specify a Windows absolute path that includes the drive letter. The *`workDir`* value will be valid only when `SCE_KERNEL_DECI4P_HOST_PROCESS_WORKDIR_OVERRIDE` is specified for *`flags`*. In other cases, specify NULL.

In addition, the string length that can be specified for *`pathName`*, *`cmdLine`*, and *`workDir`* is up to `SCE_KERNEL_DECI4P_HOST_PROCESS_MAX_STR_SIZE` bytes including the NULL terminator.

See Also

`sceKernelDeci4pCreateHostProcess()`, `sceKernelDeci4pCreateHostProcessAndWait()`

SceKernelDeci4pCreateHostProcessResult

Startup result for application in development host computer

Definition

```
#include <libdeci4p.h>
typedef struct SceKernelDeci4pCreateHostProcessResult {
        SceUInt32 hostProcessId;
        SceUInt32 hostErrorCode;
        SceUInt32 reserved0[6];
} SceKernelDeci4pCreateHostProcessResult;
```

Members

hostProcessId	Process ID in development host computer
hostErrorCode	Error code for `CreateProcessW()` called in development host computer
reserved0	Reserved area (always 0)

Description

This structure represents the startup result for an application in a development host computer. When executing an application with `sceKernelDeci4pCreateHostProcess()` or `sceKernelDeci4pCreateHostProcessAndWait()`, the startup result will be stored in this structure.

In *hostProcessId*, the process ID will be stored if the application successfully starts.

In *hostErrorCode*, if `sceKernelDeci4pCreateHostProcess()` or `sceKernelDeci4pCreateHostProcessAndWait()` returns `SCE_KERNEL_ERROR_DECI4P_HOST_CREATE_PROCESS`, an error code for the `CreateProcessW()` called in the development host computer will be stored.

See Also

`sceKernelDeci4pCreateHostProcess()`, `sceKernelDeci4pCreateHostProcessAndWait()`

SceKernelDeci4pHostProcessExitCallback

Callback function prototype for application exit in development host computer

Definition

```
#include <libdeci4p.h>
typedef SceInt32 (*SceKernelDeci4pHostProcessExitCallback)(
        SceUID notifyId,
        SceInt32 notifyCount,
        SceInt32 notifyArg,
        void *pCommon
);
```

Arguments

notifyId	`SCE_UID_INVALID_UID`
notifyCount	Notify count
notifyArg	Pointer to `SceKernelDeci4pHostProcessExitInfo` structure
pCommon	Application defined callback parameters registered with `sceKernelCreateCallback()`

Return Values

Value	Description
0	Do not delete this callback
Not 0	Delete this callback

Description

This is a prototype of the callback function that receives exit notifications for applications started with `sceKernelDeci4pCreateHostProcess()`.

To *notifyArg*, a pointer to the `SceKernelDeci4pHostProcessExitInfo` structure that represents the exit wait result will be passed (casting is required since it is passed as a `SceInt32` type).

See Also

`SceKernelDeci4pHostProcessExitInfo`, `sceKernelDeci4pCreateHostProcess()`, `sceKernelCreateCallback()`

SceKernelDeci4pHostProcessExitInfo

Exit information for application in development host computer

Definition

```
#include <libdeci4p.h>
typedef struct SceKernelDeci4pHostProcessExitInfo {
        SceInt32 result;
        SceUInt32 hostProcessId;
        SceUInt32 hostProcessExitCode;
        SceUInt32 reserved0[5];
} SceKernelDeci4pHostProcessExitInfo;
```

Members

result	Application exit wait result
hostProcessId	Process ID in development host computer
hostProcessExitCode	Application exit code
reserved0	Reserved area (always 0)

Description

This structure represents the exit wait result for an application in a development host computer. A pointer to this structure will be passed to *notifyArg* in a callback function (casting is required since it is passed as a SceInt32 type)

SCE_OK (=0) will be stored in *result* when the exit wait completed successfully. One of the following error codes (a negative value) will be stored for errors.

Value	(Number)	Description
SCE_KERNEL_ERROR_DECI4P_UNKNOWN	0x80080800	Undefined error not listed below
SCE_KERNEL_ERROR_DECI4P_INTERRUPTED	0x8008080E	Disconnected during exit wait

In *hostProcessId*, the process ID in the development host computer will be stored.

In *hostProcessExitCode*, the application exit code will be stored. This field will be valid only when *result* is SCE_OK (=0).

See Also

SceKernelDeci4pHostProcessExitInfo, sceKernelDeci4pCreateHostProcess()

sceKernelDeci4pCreateHostProcess

Asynchronously execute application in development host computer

Definition

```
#include <libdeci4p.h>
SceInt32 sceKernelDeci4pCreateHostProcess(
        const SceKernelDeci4pCreateHostProcessParam *param,
        SceKernelDeci4pCreateHostProcessResult *result,
        SceUID cbid,
        SceKernelDeci4pHostProcessExitInfo *exitInfo
);
```

Arguments

param	Application startup parameters
result	Destination to store the application startup result
cbid	Callback ID to receive exit notification
exitInfo	Destination to store application exit result

Return Values

Returns `SCE_OK` (=0) and stores the startup result in `*result` when application startup is successful. Afterward, the exit result will be stored in `*exitInfo`, and notification to `cbid` will be performed.

One of the following error codes (a negative value) will return when application startup fails. Notification to `cbid` will not be performed.

Value	(Number)	Description
SCE_KERNEL_ERROR_DECI4P_UNKNOWN	0x80080800	Undefined error not listed below
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_CBID	0x80080803	Invalid callback ID
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_ADDR	0x80080804	Invalid address
SCE_KERNEL_ERROR_DECI4P_NO_MEMORY	0x80080807	Insufficient memory
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_RESERVED	0x80080808	Reserved area is being used
SCE_KERNEL_ERROR_DECI4P_NO_CONNECT	0x8008080A	Not connected to development host computer
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_PARAM	0x8008080D	Invalid startup parameter
SCE_KERNEL_ERROR_DECI4P_HOST_PROCESS_DISABLED	0x8008080F	This feature is disabled in development host computer
SCE_KERNEL_ERROR_DECI4P_HOST_CREATE_PROCESS	0x80080810	Process creation in development host computer failed
SCE_KERNEL_ERROR_DECI4P_HOST_TOOMANY_PROCESS	0x80080811	Number of processes being generated is too many

Description

This function asynchronously executes an application in a development host computer. This function will return immediately after application startup without waiting for it to exit. When the application exits, the specified callback will be notified.

For *`param`*, specify the application startup parameters with an `SceKernelDeci4pCreateHostProcessParam` structure. The startup result will be stored in *`result`*.

For *`cbid`*, specify the ID of the callback created with the `sceKernelCreateCallback()` function.

If this function is successful, notification will be performed when the application exits or when an error occurs while waiting for exit. At this time, the exit result or error code will be stored in `*`*`exitInfo`*. Make sure that the memory area pointed to by *`exitInfo`* is not invalid until notification to the callback is performed when this function is successful (particularly when using local variables).

Notes

Application startup is performed by Target Manager Server calling the Win32 API `CreateProcessW()` in a development host computer.

See Also

`SceKernelDeci4pCreateHostProcessParam`, `SceKernelDeci4pCreateHostProcessResult`, `SceKernelDeci4pHostProcessExitCallback`, `SceKernelDeci4pHostProcessExitInfo`

sceKernelDeci4pCreateHostProcessAndWait

Execute application in development host computer and wait until exit

Definition

```
#include <libdeci4p.h>
SceInt32 sceKernelDeci4pCreateHostProcessAndWait(
        const SceKernelDeci4pCreateHostProcessParam *param,
        SceKernelDeci4pCreateHostProcessResult *result,
        SceUInt32 *hostProcessExitCode
);
```

Arguments

param	Application startup parameters
result	Destination to store the application startup result
hostProcessExitCode	Destination to store the application exit code

Return Values

Stores the startup result in **result* when application startup is successful.

Returns one of the following error codes (a negative value) when the application could not start due to an error.

Value	(Number)	Description
SCE_KERNEL_ERROR_DECI4P_UNKNOWN	0x80080800	Undefined error not listed below
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_CBID	0x80080803	Invalid callback ID
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_ADDR	0x80080804	Invalid address
SCE_KERNEL_ERROR_DECI4P_NO_MEMORY	0x80080807	Insufficient memory
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_RESERVED	0x80080808	Reserved area is being used
SCE_KERNEL_ERROR_DECI4P_NO_CONNECT	0x8008080A	Not connected to development host computer
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_PARAM	0x8008080D	Invalid startup parameter
SCE_KERNEL_ERROR_DECI4P_HOST_PROCESS_DISABLED	0x8008080F	Functionality in development host computer is disabled
SCE_KERNEL_ERROR_DECI4P_HOST_CREATE_PROCESS	0x80080810	Process creation in development host computer failed
SCE_KERNEL_ERROR_DECI4P_HOST_TOOMANY_PROCESS	0x80080811	Number of processes being generated is too many

The exit code will be stored in **hostProcessExitCode* for completion up to application exit.

This function will return `SCE_OK` (=0) when up to application exit is successful.

One of the following error codes (a negative value) will return when an error occurred during the wait for the application to exit after application startup.

Value	(Number)	Description
SCE_KERNEL_ERROR_DECI4P_UNKNOWN	0x80080800	Undefined error not listed below
SCE_KERNEL_ERROR_DECI4P_INTERRUPTED	0x8008080E	Disconnected during exit wait

Description

This function executes an application in a development host computer. When application startup is successful, this function will block the caller thread until the application exits.

For *`param`*, specify the application startup parameters with an `SceKernelDeci4pCreateHostProcessParam` structure.

The startup result will be stored in *`*result`*, and the exit code will be stored in *`*hostProcessExitCode`*. For details on application startup results, refer to the "`SceKernelDeci4pCreateHostProcessResult`" section.

Notes

Application startup is performed by Target Manager Server calling the Win32 API `CreateProcessW()` in a development host computer.

See Also

`SceKernelDeci4pCreateHostProcessParam`, `SceKernelDeci4pCreateHostProcessResult`

Constants

List of libdeci4p constants

Definition

Value	(Number)	Description
`SCE_KERNEL_DECI4P_CALLBACKARG_DATA_READY`	0x10000000	Data received
`SCE_KERNEL_DECI4P_CALLBACKARG_NOCONNECT`	0x20000000	Not connected to development host computer
`SCE_KERNEL_DECI4P_CALLBACKARG_NOPROTOCOL`	0x40000000	Protocol not registered
`SCE_KERNEL_DECI4P_HOST_PROCESS_WINDOW_SHOW`	0x00000000	Show window
`SCE_KERNEL_DECI4P_HOST_PROCESS_WINDOW_HIDE`	0x00000001	Hide window
`SCE_KERNEL_DECI4P_HOST_PROCESS_WORKDIR_FSROOT`	0x00000000	Make current directory file server directory
`SCE_KERNEL_DECI4P_HOST_PROCESS_WORKDIR_EXECUTABLE`	0x00000002	Make current directory with executable file
`SCE_KERNEL_DECI4P_HOST_PROCESS_WORKDIR_OVERRIDE`	0x00000004	Specify current directory
`SCE_KERNEL_DECI4P_HOST_PROCESS_MAX_STR_SIZE`	1024	Maximum string length (including the NULL terminator)

Return Codes

List of return codes returned by libdeci4p functions

Definition

Value	(Number)	Description
SCE_KERNEL_ERROR_DECI4P_UNKNOWN	0x80080800	Undefined error not listed below
SCE_KERNEL_ERROR_DECI4P_ALREADYUSE_PROTOCOL	0x80080801	Protocol number that is already used
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_SID	0x80080802	Invalid socket ID
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_CBID	0x80080803	Invalid callback ID
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_ADDR	0x80080804	Invalid address
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_SIZE	0x80080805	Invalid size
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_PROTOCOL	0x80080806	Invalid protocol number
SCE_KERNEL_ERROR_DECI4P_NO_MEMORY	0x80080807	Insufficient memory
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_RESERVED	0x80080808	Invalid value
SCE_KERNEL_ERROR_DECI4P_TOOMANY_PROTOCOL	0x80080809	Too many protocols
SCE_KERNEL_ERROR_DECI4P_NO_CONNECT	0x8008080A	Not connected to development host computer
SCE_KERNEL_ERROR_DECI4P_NO_PROTO	0x8008080B	Protocol not registered in Target Manager API
SCE_KERNEL_ERROR_DECI4P_TOOSMALL_BUFFER	0x8008080C	Buffer is too small
SCE_KERNEL_ERROR_DECI4P_ILLEGAL_PARAM	0x8008080D	Invalid startup parameter
SCE_KERNEL_ERROR_DECI4P_INTERRUPTED	0x8008080E	Interrupted during exit wait
SCE_KERNEL_ERROR_DECI4P_HOST_PROCESS_DISABLED	0x8008080F	Functionality in development host computer is disabled
SCE_KERNEL_ERROR_DECI4P_HOST_CREATE_PROCESS	0x80080810	Process creation in development host computer failed
SCE_KERNEL_ERROR_DECI4P_HOST_TOOMANY_PROCESS	0x80080811	Number of processes being generated is too many