libfios2 Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

Defines

Define Summary

Define	Value	Description
SCE_FIOS_BUFFER_INITIALIZER	{ 0, 0 }	Initializes SceFiosBuffer to default values.
SCE_FIOS_CALLBACK_THREAD_STACKSIZE	(8*1024)	Stack size for callback threads (PlayStation®Vita).
SCE_FIOS_CALLBACK_THREAD_STACKSIZE	(16*1024)	Stack size for callback threads (Windows).
SCE_FIOS_CHUNK_DEFAULT	(256*1024)	Default chunk size for large I/O requests.
SCE_FIOS_CHUNK_SIZE	64	Average estimated chunk size.
SCE_FIOS_CHUNK_STORAGE_ALIGNMENT	8	Minimum alignment required for the SceFiosParams.*chunkStorage* buffer.
SCE_FIOS_CHUNK_STORAGE_SIZE_MAX	SCE_FIOS_STORAGE_SIZE(SCE_FIOS_MAX_ALLOCATION_SIZE, SCE_FIOS_CHUNK_ALLOCATION_UNIT)	Maximum supported size for the SceFiosParams.*chunkStorage* buffer.
SCE_FIOS_DECOMPRESSOR_THREAD_COUNT_DEFAULT	2	The default number of decompression threads that will be created if no value has been specified.
SCE_FIOS_DECOMPRESSOR_THREAD_COUNT_MAX	3	The maximum number of decompression threads that can be created.
SCE_FIOS_DECOMPRESSOR_THREAD_DEFAULT_AFFINITY	SCE_KERNEL_THREAD_CPU_AFFINITY_MASK_DEFAULT	Default decompressor thread affinity.
SCE_FIOS_DECOMPRESSOR_THREAD_DEFAULT_PRIORITY	(SCE_KERNEL_LOWEST_PRIORITY_USER-2)	Default priority for decompressor threads (PlayStation®Vita).
SCE_FIOS_DECOMPRESSOR_THREAD_DEFAULT_PRIORITY	(THREAD_PRIORITY_BELOW_NORMAL)	Default priority for decompressor threads (Windows).
SCE_FIOS_DECOMPRESSOR_THREAD_STACKSIZE	(12*1024)	Stack size for decompressor threads (PlayStation®Vita).
SCE_FIOS_DECOMPRESSOR_THREAD_STACKSIZE	(16*1024)	Default stack size for decompressor threads (Windows).
SCE_FIOS_DH_INVALID	0	An invalid SceFiosDH value which can be used for initialization.
SCE_FIOS_DH_SIZE	80	Size of DH data structure.
SCE_FIOS_DH_STORAGE_ALIGNMENT	8	Minimum alignment required for the SceFiosParams.*dhStorage* buffer (PlayStation®Vita and 64-bit Windows).
SCE_FIOS_DH_STORAGE_ALIGNMENT	4	Minimum alignment required for the SceFiosParams.*dhStorage* buffer (32-bit Windows).
SCE_FIOS_DIRENTRY_INITIALIZER	{ 0, 0, 0, 0, 0, {0,0,0}, "" }	Initializes SceFiosDirEntry to default values.
SCE_FIOS_ERROR_ACCESS	-2138963949	Insufficient access privileges. (0x80820013)
SCE_FIOS_ERROR_ALREADY_EXISTS	-2138963933	The file or directory already exists. (0x80820023)
SCE_FIOS_ERROR_BAD_ALIGNMENT	-2138963955	Invalid alignment on a pointer argument. (0x8082000D)

Define	Value	Description
SCE_FIOS_ERROR_BAD_ARCHIVE	-2138963937	Badly-formed or unsupported PSARC archive. (0x8082001F)
SCE_FIOS_ERROR_BAD_DH	-2138963956	An invalid SceFiosDH was given as an argument. (0x8082000C)
SCE_FIOS_ERROR_BAD_FH	-2138963957	An invalid SceFiosFH was given as an argument. (0x8082000B)
SCE_FIOS_ERROR_BAD_FLAGS	-2138963935	Invalid flags were given as an argument. (0x80820021)
SCE_FIOS_ERROR_BAD_INDEX	-2138963940	An invalid index was given. Either out of range, or already in use. (0x8082001C)
SCE_FIOS_ERROR_BAD_IOVCNT	-2138963959	An invalid iovcnt was given as an argument. (0x80820009)
SCE_FIOS_ERROR_BAD_OFFSET	-2138963961	An invalid offset was given as an argument. (0x80820007)
SCE_FIOS_ERROR_BAD_OP	-2138963958	An invalid SceFiosOp was given as an argument. (0x8082000A)
SCE_FIOS_ERROR_BAD_ORDER	-2138963941	An invalid order was given as an argument. (0x8082001B)
SCE_FIOS_ERROR_BAD_OVERLAY	-2138963942	An invalid overlay was given as an argument. (0x8082001A)
SCE_FIOS_ERROR_BAD_PATH	-2138963963	File not found. (0x80820005)
SCE_FIOS_ERROR_BAD_PTR	-2138963962	An invalid pointer was given as an argument. (0x80820006)
SCE_FIOS_ERROR_BAD_RESOLVE_TYPE	-2138963936	An invalid Resolve Type was given when resolving overlays. (0x80820020)
SCE_FIOS_ERROR_BAD_SIZE	-2138963960	An invalid size was given as an argument. (0x80820008)
SCE_FIOS_ERROR_BUSY	-2138963938	A resource is busy; try again later. (0x8082001E)
SCE_FIOS_ERROR_CANCELLED	-2138963950	Operation was cancelled. (0x80820012)
SCE_FIOS_ERROR_CANT_ALLOCATE_CHUNK	-2138963964	Out of memory in *chunkStorage*. (0x80820004)
SCE_FIOS_ERROR_CANT_ALLOCATE_DH	-2138963965	Out of memory in *dhStorage*. (0x80820003)
SCE_FIOS_ERROR_CANT_ALLOCATE_FH	-2138963966	Out of memory in *fhStorage*. (0x80820002)
SCE_FIOS_ERROR_CANT_ALLOCATE_OP	-2138963967	Out of memory in *opStorage*. (0x80820001)
SCE_FIOS_ERROR_DECOMPRESSION	-2138963948	Decompression failed. (0x80820014)
SCE_FIOS_ERROR_EOF	-2138963952	End-of-file reached. (0x80820010)
SCE_FIOS_ERROR_EVENT_NOT_HANDLED	-2138963939	Callback did not handle the event. (0x8082001D)
SCE_FIOS_ERROR_MEDIA_GONE	-2138963945	Media has been removed, unplugged, or otherwise detached. (0x80820017)
SCE_FIOS_ERROR_NOT_A_DIRECTORY	-2138963953	Attempted a directory operation, but target was a file. (0x8082000F)
SCE_FIOS_ERROR_NOT_A_FILE	-2138963954	Attempted a file operation, but target was a directory. (0x8082000E)
SCE_FIOS_ERROR_PATH_TOO_LONG	-2138963944	Path does not fit in buffer. (0x80820018)
SCE_FIOS_ERROR_READ_ONLY	-2138963947	Attempted to write to read-only filehandle or media. (0x80820015)
SCE_FIOS_ERROR_TIMEOUT	-2138963951	Timeout occurred. (0x80820011)

Define	Value	Description
SCE_FIOS_ERROR_TOO_MANY_OVERLAYS	-2138963943	Too many overlays. (0x80820019)
SCE_FIOS_ERROR_UNIMPLEMENTED	-2138963968	Not implemented. (0x80820000)
SCE_FIOS_ERROR_UNKNOWN	-2138963934	An unknown, platform-specific error occurred. (0x80820022)
SCE_FIOS_ERROR_WRITE_ONLY	-2138963946	Attempted to read from write-only filehandle. (0x80820016)
SCE_FIOS_FH_INVALID	0	An invalid SceFiosFH value which can be used for initialization.
SCE_FIOS_FH_SIZE	80	Size of FH data structure.
SCE_FIOS_FH_STORAGE_ALIGNMENT	8	Minimum alignment required for the SceFiosParams.*fhStorage* buffer (PlayStation®Vita and 64-bit Windows).
SCE_FIOS_FH_STORAGE_ALIGNMENT	4	Minimum alignment required for the SceFiosParams.*fhStorage* buffer (32-bit Windows).
SCE_FIOS_HANDLE_INVALID	0	An invalid SceFiosHandle value which can be used for initialization.
SCE_FIOS_IN_PROGRESS	1	The operation has not completed yet.
SCE_FIOS_INVALID_LBA	0x7FFFFFFF FFFFFFFFLL	Invalid LBA value.
SCE_FIOS_IO_THREAD_DEFAULT_AFFINITY	SCE_FIOS_THREAD_DEFAULT_AFFINITY	Default I/O thread affinity.
SCE_FIOS_IO_THREAD_DEFAULT_PRIORITY	(SCE_KERNEL_HIGHEST_PRIORITY_USER+2)	Default priority for I/O threads (PlayStation®Vita).
SCE_FIOS_IO_THREAD_DEFAULT_PRIORITY	(THREAD_PRIORITY_TIME_CRITICAL)	Default priority for I/O threads (Windows).
SCE_FIOS_IO_THREAD_STACKSIZE	(8*1024)	Stack size for I/O threads (PlayStation®Vita).
SCE_FIOS_IO_THREAD_STACKSIZE	(16*1024)	Default stack size for I/O threads (Windows).
SCE_FIOS_MICROSECONDS_PER_MILLISECOND	1000LL	Number of microseconds in a millisecond.
SCE_FIOS_MICROSECONDS_PER_SECOND	1000000LL	Number of microseconds in a second.
SCE_FIOS_MILLISECONDS_PER_SECOND	1000LL	Number of milliseconds in a second.
SCE_FIOS_NANOSECONDS_PER_MICROSECOND	1000LL	Number of nanoseconds in a microsecond.
SCE_FIOS_NANOSECONDS_PER_MILLISECOND	1000000LL	Number of nanoseconds in a millisecond.
SCE_FIOS_NANOSECONDS_PER_SECOND	1000000000LL	Number of nanoseconds in a second.
SCE_FIOS_OFFSET_MAX	((int64_t) 0x7FFFFFFF FFFFFFFFLL)	Maximum value of a SceFiosOffset.
SCE_FIOS_OK	0	Success.
SCE_FIOS_OP_INVALID	0	An invalid SceFiosOp value which can be used for initialization.
SCE_FIOS_OP_SIZE	168	Size of OP data structure.
SCE_FIOS_OP_STORAGE_ALIGNMENT	8	Minimum alignment required for the SceFiosParams.*opStorage* buffer (PlayStation®Vita and 64-bit Windows).

Define	Value	Description
SCE_FIOS_OP_STORAGE_ALIGNMENT	4	Minimum alignment required for the SceFiosParams.*opStorage* buffer (32-bit Windows).
SCE_FIOS_OPATTR_INITIALIZER	{ 0, 0, 0, 0, 0, 0, 0, 0 }	Initializes SceFiosOpAttr to default values.
SCE_FIOS_OPENPARAMS_ INITIALIZER	{ 0, 0, 0, SCE_FIOS_BUFFER_ INITIALIZER }	Initializes SceFiosOpenParams to default values.
SCE_FIOS_OVERLAY_INITIALIZER	{0,0,{0,0,0,0, 0,0,0,0,0,0}, 0,"","",}	Initializes SceFiosOverlay to default values.
SCE_FIOS_PARAMS_ INITIALIZER	{ 0, sizeof(SceFiosParams), 0, 0, \ SCE_FIOS_IO_ THREAD_COUNT_MAX, SCE_FIOS_SCHEDULER_ THREAD_COUNT_ DEFAULT, 0, SCE_FIOS_ CHUNK_DEFAULT, \ SCE_FIOS_ DECOMPRESSOR_ THREAD_COUNT_ DEFAULT, 0, 0, 0, 0, 0, \ SCE_FIOS_BUFFER_ INITIALIZER, SCE_FIOS_BUFFER_ INITIALIZER, SCE_FIOS_BUFFER_ INITIALIZER, SCE_FIOS_BUFFER_ INITIALIZER, \ NULL, NULL, NULL, \ { SCE_FIOS_ IO_THREAD_ DEFAULT_PRIORITY, SCE_FIOS_ DECOMPRESSOR_ THREAD_DEFAULT_ PRIORITY }, \ { SCE_FIOS_ IO_THREAD_ DEFAULT_AFFINITY, SCE_FIOS_ DECOMPRESSOR_ THREAD_DEFAULT_ AFFINITY}}	Initializes SceFiosParams to default values.
SCE_FIOS_PATH_DEFAULT	1024	Default length used for paths in FIOS objects.
SCE_FIOS_PATH_MAX	1024	Maximum path length.
SCE_FIOS_PRIO_DEFAULT	((int8_t)0)	Default priority. See SceFiosPriority.
SCE_FIOS_PRIO_MAX	((int8_t)127)	Maximum priority. See SceFiosPriority.
SCE_FIOS_PRIO_MIN	((int8_t)-128)	Minimum priority. See SceFiosPriority.

Define	Value	Description
SCE_FIOS_PSARC_DEARCHIVER_CONTEXT_INITIALIZER	{ sizeof(SceFiosPsarcDearchiverContext), 0, 0, 0, {0,0,0} }	Initializer for the PSARC dearchiver context.
SCE_FIOS_PSARC_DEARCHIVER_TEMP_BUFFERS	3	The number of temporary buffers used.
SCE_FIOS_PSARC_DEARCHIVER_WORK_BUFFER_ALIGNMENT	64	The minimum alignment of the work buffer for SceFiosPsarcDearchiverContext.
SCE_FIOS_PSARC_DEARCHIVER_WORK_BUFFER_SIZE	(3*64*1024)	The default size of the work buffer for SceFiosPsarcDearchiverContext.
SCE_FIOS_RAM_CACHE_BUFFER_ALIGNMENT	8	The minimum alignment of the work for buffer SceFiosRamCacheContext.
SCE_FIOS_RAM_CACHE_BUFFER_SIZE	(128*1024)	Deprecated macro to specify the minimum size of the work buffer for SceFiosRamCacheContext.
SCE_FIOS_RAM_CACHE_CONTEXT_INITIALIZER	{ sizeof(SceFiosRamCacheContext), 0, (64 * 1024), NULL, NULL, 0, {0,0,0} }	Initializer for the RAM cache context.
SCE_FIOS_SECONDS_PER_DAY	86400LL	Number of seconds in a day.
SCE_FIOS_SECONDS_PER_HOUR	3600LL	Number of seconds in an hour.
SCE_FIOS_SECONDS_PER_MINUTE	60LL	Number of seconds in a minute.
SCE_FIOS_SECONDS_PER_YEAR	31557600LL	Approximate number of seconds in a year.
SCE_FIOS_STAT_INITIALIZER	{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }	Initializes SceFiosStat to default values.
SCE_FIOS_THREAD_DEFAULT_AFFINITY	SCE_KERNEL_CPU_MASK_USER_2	Default thread affinity.
SCE_FIOS_TIME_EARLIEST	((SceFiosTime)1)	Earliest possible time value. See SceFiosTime.
SCE_FIOS_TIME_LATEST	((SceFiosTime) 0x7FFFFFFF FFFFFFFFLL)	Latest possible time value. See SceFiosTime.
SCE_FIOS_TIME_NULL	((SceFiosTime)0)	Special time value meaning *undefined*. See SceFiosTime.
SCE_FIOS_TUPLE_INITIALIZER	{ 0, 0, "" }	Initializes SceFiosTuple to default values.

Macros

SCE_FIOS_CHUNK_STORAGE_SIZE

Size required for the SceFiosParams.*chunkStorage* buffer.

Definition

```
#include <fios2_types.h>
#define SCE_FIOS_CHUNK_STORAGE_SIZE(
    numChunks
) SCE_FIOS_STORAGE_SIZE(numChunks, SCE_FIOS_CHUNK_SIZE)
```

Arguments

[in] *numChunks* Minimum number of chunks required.

Description

Size required for the SceFiosParams.*chunkStorage* buffer. The size of each chunk varies, so SCE_FIOS_STORAGE_SIZE returns an approximate size required based on *numChunks*.

SCE_FIOS_DH_STORAGE_SIZE

Size required for the SceFiosParams.*dhStorage* buffer.

Definition

```
#include <fios2_types.h>
#define SCE_FIOS_DH_STORAGE_SIZE(
    numDHs,
    pathMax
) SCE_FIOS_STORAGE_SIZE(numDHs, SCE_FIOS_DH_SIZE + pathMax)
```

Arguments

[in] *numDHs*	Minimum number of DHs required. Using a value higher than SCE_FIOS_MAX_HANDLE_ELEMENTS will cause sceFiosInitialize to fail.
[in] *pathMax*	Value of SceFiosParams.*pathMax*, or SCE_FIOS_PATH_DEFAULT, or SCE_FIOS_PATH_MAX.

Description

Size required for the SceFiosParams.*dhStorage* buffer.

SCE_FIOS_FH_STORAGE_SIZE

Size required for the SceFiosParams.*fhStorage* buffer.

Definition

```
#include <fios2_types.h>
#define SCE_FIOS_FH_STORAGE_SIZE(
    numFHs,
    pathMax
) SCE_FIOS_STORAGE_SIZE(numFHs, SCE_FIOS_FH_SIZE + pathMax)
```

Arguments

[in] *numFHs*	Minimum number of FHs required. Using a value higher than SCE_FIOS_MAX_HANDLE_ELEMENTS will cause sceFiosInitialize to fail.
[in] *pathMax*	Value of SceFiosParams.*pathMax*, or SCE_FIOS_PATH_DEFAULT, or SCE_FIOS_PATH_MAX.

Description

Size required for the SceFiosParams.*fhStorage* buffer.

SCE_FIOS_OP_STORAGE_SIZE

Size required for the SceFiosParams.*opStorage* buffer.

Definition

```
#include <fios2_types.h>
#define SCE_FIOS_OP_STORAGE_SIZE(
    numOps,
    pathMax
) SCE_FIOS_STORAGE_SIZE(numOps, SCE_FIOS_OP_SIZE + pathMax)
```

Arguments

[in] *numOps*	Minimum number of ops required. Using a value higher than SCE_FIOS_MAX_HANDLE_ELEMENTS will cause sceFiosInitialize to fail.
[in] *pathMax*	Value of SceFiosParams.*pathMax*, or SCE_FIOS_PATH_DEFAULT, or SCE_FIOS_PATH_MAX.

Description

Size required for the SceFiosParams.*opStorage* buffer.

Document serial number: 000004892117

SCE_FIOS_PSARC_DEARCHIVER_WORK_BUFFER

The size of the work buffer for SceFiosPsarcDearchiverContext based on the maximum block size of all mounted PSARC files.

Definition

```
#include <fios2_filters.h>
#define SCE_FIOS_PSARC_DEARCHIVER_WORK_BUFFER(blockSize) ((blockSize) *
SCE_FIOS_PSARC_DEARCHIVER_TEMP_BUFFERS)
```

Arguments

None

Description

The size of the work buffer for SceFiosPsarcDearchiverContext based on the maximum block size of all mounted PSARC files.

SCE_FIOS_RAM_CACHE_BUFFER_SIZE_PER_BLOCK

Macro to determine size of the work buffer for SceFiosRamCacheContext given a block size and block count.

Definition

```
#include <fios2_filters.h>
#define SCE_FIOS_RAM_CACHE_BUFFER_SIZE_PER_BLOCK(
    blocks,
    blocksize,
    pathMax
) ((blocks * (blocksize + 64)) + SCE_FIOS_ALIGN_UP((72 + pathMax + 1),8))
```

Arguments

[in] *blocks*	Number of cache blocks desired.
[in] *blocksize*	Size in bytes of each cache block.
[in] *pathMax*	Value of SceFiosParams.*pathMax*, or SCE_FIOS_PATH_DEFAULT, or SCE_FIOS_PATH_MAX.

Description

Macro to determine size of the work buffer for SceFiosRamCacheContext given a block size and block count.

sceFiosTimeIntervalFromMicroseconds

Converts microseconds to a SceFiosTimeInterval.

Definition

```
#include <fios2_api.h>
#define sceFiosTimeIntervalFromMicroseconds(
    us
) sceFiosTimeIntervalFromNanoseconds((us) *
SCE_FIOS_NANOSECONDS_PER_MICROSECOND)
```

Arguments

[in] *us* Microseconds to convert.

Description

Converts microseconds to a SceFiosTimeInterval.

Notes

Result is the equivalent SceFiosTimeInterval.

sceFiosTimeIntervalFromMilliseconds

Converts milliseconds to a SceFiosTimeInterval.

Definition

```
#include <fios2_api.h>
#define sceFiosTimeIntervalFromMilliseconds(
    ms
) sceFiosTimeIntervalFromNanoseconds((ms) *
SCE_FIOS_NANOSECONDS_PER_MILLISECOND)
```

Arguments

[in] *ms* Milliseconds to convert.

Description

Converts milliseconds to a SceFiosTimeInterval.

Notes

Result is the equivalent SceFiosTimeInterval.

sceFiosTimeIntervalFromSeconds

Converts seconds to a SceFiosTimeInterval.

Definition

```
#include <fios2_api.h>
#define sceFiosTimeIntervalFromSeconds(
    s
) sceFiosTimeIntervalFromNanoseconds((s) * SCE_FIOS_NANOSECONDS_PER_SECOND)
```

Arguments

[in] *s* Seconds to convert.

Description

Converts seconds to a SceFiosTimeInterval.

Notes

Result is the equivalent SceFiosTimeInterval.

sceFiosTimeIntervalToMicroseconds

Converts a SceFiosTimeInterval to microseconds.

Definition

```
#include <fios2_api.h>
#define sceFiosTimeIntervalToMicroseconds(
    interval
) SCE_FIOS_DIVIDE_ROUNDING_UP(sceFiosTimeIntervalToNanoseconds(interval),
SCE_FIOS_NANOSECONDS_PER_MICROSECOND)
```

Arguments

[in] *interval* Interval to convert.

Description

Converts a SceFiosTimeInterval to microseconds.

Notes

The result is rounded upward, so an interval representing 1.5 microseconds will return the value 2.

Result is the equivalent number of microseconds.

sceFiosTimeIntervalToMilliseconds

Converts a SceFiosTimeInterval to milliseconds.

Definition

```
#include <fios2_api.h>
#define sceFiosTimeIntervalToMilliseconds(
    interval
) SCE_FIOS_DIVIDE_ROUNDING_UP(sceFiosTimeIntervalToNanoseconds(interval),
SCE_FIOS_NANOSECONDS_PER_MILLISECOND)
```

Arguments

[in] *interval* Interval to convert.

Description

Converts a SceFiosTimeInterval to milliseconds.

Notes

The result is rounded upward, so an interval representing 1.5 milliseconds will return the value 2.

Result is the equivalent number of milliseconds.

sceFiosTimeIntervalToSeconds

Converts a SceFiosTimeInterval to seconds.

Definition

```
#include <fios2_api.h>
#define sceFiosTimeIntervalToSeconds(
    interval
) SCE_FIOS_DIVIDE_ROUNDING_UP(sceFiosTimeIntervalToNanoseconds(interval),
SCE_FIOS_NANOSECONDS_PER_SECOND)
```

Arguments

[in] *interval* Interval to convert.

Description

Converts a SceFiosTimeInterval to seconds.

Notes

The result is rounded upward, so an interval representing 1.5 seconds will return the value 2.

Result is the equivalent number of seconds.

sceFiosTimeoutElapsed

Checks for a timeout condition.

Definition

```
#include <fios2_api.h>
#define sceFiosTimeoutElapsed(
    start,
    timeout
) ((sceFiosTimeGetCurrent() - start) > timeout)
```

Arguments

[in] *start* The starting point.
[in] *timeout* The timeout interval.

Description

Checks for a timeout condition. Returns true if the timeout interval has elapsed.

sceFiosTimeRelativeMicroseconds

Returns a SceFiosTime representing a number of microseconds from now.

Definition

```
#include <fios2_api.h>
#define sceFiosTimeRelativeMicroseconds(
    us
) (sceFiosTimeGetCurrent() + sceFiosTimeIntervalFromMicroseconds(us))
```

Arguments

[in] *us* Microseconds from now.

Description

Returns a SceFiosTime representing a number of microseconds from now.

Notes

Result is the equivalent SceFiosTime.

sceFiosTimeRelativeMilliseconds

Returns a SceFiosTime representing a number of milliseconds from now.

Definition

```
#include <fios2_api.h>
#define sceFiosTimeRelativeMilliseconds(
    ms
) (sceFiosTimeGetCurrent() + sceFiosTimeIntervalFromMilliseconds(ms))
```

Arguments

[in] *ms* Milliseconds from now.

Description

Returns a SceFiosTime representing a number of milliseconds from now.

Notes

Result is the equivalent SceFiosTime.

sceFiosTimeRelativeNanoseconds

Returns a SceFiosTime representing a number of nanoseconds from now.

Definition

```
#include <fios2_api.h>
#define sceFiosTimeRelativeNanoseconds(
    ns
) (sceFiosTimeGetCurrent() + sceFiosTimeIntervalFromNanoseconds(ns))
```

Arguments

[in] *ns* Nanoseconds from now.

Description

Returns a SceFiosTime representing a number of nanoseconds from now.

Notes

Result is the equivalent SceFiosTime.

sceFiosTimeRelativeSeconds

Returns a SceFiosTime representing a number of seconds from now.

Definition

```
#include <fios2_api.h>
#define sceFiosTimeRelativeSeconds(
    s
) (sceFiosTimeGetCurrent() + sceFiosTimeIntervalFromSeconds(s))
```

Arguments

[in] *s* Seconds from now.

Description

Returns a SceFiosTime representing a number of seconds from now.

Notes

Result is the equivalent SceFiosTime.

Functions

sceFiosArchiveGetDecompressorThreadCount

Get the number of decompression threads currently in use.

Definition

```
#include <fios2_api.h>
uint8_t sceFiosArchiveGetDecompressorThreadCount();
```

Arguments

None

Return Values

The number of threads currently in use.

Description

Get the number of decompression threads currently in use.

See Also

SceFiosParams, sceFiosArchiveSetDecompressorThreadCount()

sceFiosArchiveGetMountBufferSize

Gets the size of the memory buffer required to mount an archive.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosArchiveGetMountBufferSize(
    const SceFiosOpAttr *pAttr,
    const char *pArchivePath,
    const SceFiosOpenParams *pOpenParams
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pArchivePath*	Path to an archive to query.
[in] *pOpenParams*	Open parameters. May be NULL, in which case the file will be opened with normal read-only permission.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Gets the size of the memory buffer required to mount an archive. This function will typically open the archive file, read the header, and close the file again. Upon completion, the mount buffer size will be returned via sceFiosOpGetActualCount().

Notes

A dearchiver must be present in the I/O filter stack for this function to succeed.

See Also

sceFiosIOFilterPsarcDearchiver(), sceFiosArchiveMount()

sceFiosArchiveGetMountBufferSizeSync

Gets the size of the memory buffer required to mount an archive (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosArchiveGetMountBufferSizeSync(
    const SceFiosOpAttr *pAttr,
    const char *pArchivePath,
    const SceFiosOpenParams *pOpenParams
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pArchivePath*	Path to an archive to query.
[in] *pOpenParams*	Open parameters. May be NULL, in which case the file will be opened with normal read-only permission.

Return Values

Non-negative values are the required mount buffer size. Negative values are error codes.

Description

Gets the size of the memory buffer required to mount an archive (sync). This function typically opens the archive file, reads the header, and closes the file again.

Notes

A dearchiver must be present in the I/O filter stack for this function to succeed.

See Also

sceFiosIOFilterPsarcDearchiver(), sceFiosArchiveMountSync()

sceFiosArchiveMount

Mounts an archive.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosArchiveMount(
    const SceFiosOpAttr *pAttr,
    SceFiosFH *pOutFH,
    const char *pArchivePath,
    const char *pMountPoint,
    SceFiosBuffer mountBuffer,
    const SceFiosOpenParams *pOpenParams
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[out] *pOutFH*	Returns a file handle to the archive, which can be used to unmount it later.
[in] *pArchivePath*	Path to an archive to mount.
[in] *pMountPoint*	Mount point where the archive's contents will appear.
[in] *mountBuffer*	Buffer to hold archive TOC and related data. This will be used until the archive is unmounted.
[in] *pOpenParams*	Open parameters. May be NULL, in which case the file will be opened with normal read-only permission.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Mounts an archive. After an archive is mounted, its contents will be available at the mount point.

Memory space is required to mount most archives. If you don't pass in a sufficient amount of memory in *mountBuffer* the request will fail with SCE_FIOS_ERROR_BAD_SIZE. Whether the mount succeeds or fails, the actual amount of memory required to mount the archive will be returned via sceFiosOpGetActualCount() if it can be determined.

Notes

A dearchiver must be present in the I/O filter stack for this function to succeed. See the *libfios2 Overview* document for information about how conflicts are resolved when multiple archives are mounted.

See Also

sceFiosIOFilterPsarcDearchiver(), sceFiosArchiveUnmount()

sceFiosArchiveMountSync

Mounts an archive (sync).

Definition

```
#include <fios2_api.h>
int sceFiosArchiveMountSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH *pOutFH,
    const char *pArchivePath,
    const char *pMountPoint,
    SceFiosBuffer mountBuffer,
    const SceFiosOpenParams *pOpenParams
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[out] *pOutFH*	Returns a file handle to the archive, which can be used to unmount it later.
[in] *pArchivePath*	Path to an archive to mount.
[in] *pMountPoint*	Mount point where the archive's contents will appear.
[in] *mountBuffer*	Buffer to hold archive TOC and related data. This will be used until the archive is unmounted.
[in] *pOpenParams*	Open parameters. May be NULL, in which case the file will be opened with normal read-only permission.

Return Values

SCE_FIOS_OK for success, SCE_FIOS_ERROR_BAD_SIZE if *mountBuffer* was too small, or an error code.

Description

Mounts an archive (sync). After an archive is mounted, its contents will be available at the mount point.

Memory space is required to mount most archives. If you don't pass in a sufficient amount of memory in *mountBuffer* the request will fail with SCE_FIOS_ERROR_BAD_SIZE. The actual amount of memory required to mount the archive can be determined with sceFiosArchiveGetMountBufferSizeSync().

Notes

A dearchiver must be present in the I/O filter stack for this function to succeed. Please see the *libfios2 Overview* document for information about how conflicts are resolved when multiple archives are mounted.

See Also

sceFiosIOFilterPsarcDearchiver(), sceFiosArchiveUnmountSync()

sceFiosArchiveSetDecompressorThreadCount

Set the number of decompression threads to use.

Definition

```
#include <fios2_api.h>
uint8_t sceFiosArchiveSetDecompressorThreadCount(
    uint8_t threadCount
);
```

Arguments

[in] *threadCount*	Number of threads to use. If this value is 0, one thread will be used. If this value is greater than *maxDecompressorThreadCount*, *maxDecompressorThreadCount* will be used.

Return Values

The new number of threads in use.

Description

Set the number of decompression threads to use. When a dearchiver filter is first added, it will create the number of threads specified in SceFiosParams.*maxDecompressorThreadCount*. All of these threads will persist for the lifetime of the dearchiver. You can use this function to control how many of these threads will actually be used by the decompressor.

Notes

A dearchiver must be present in the I/O filter stack for this function to have any effect.

See Also

SceFiosParams, sceFiosArchiveGetDecompressorThreadCount()

sceFiosArchiveUnmount

Unmounts a previously mounted archive.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosArchiveUnmount(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle to the archive.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Unmounts a previously mounted archive. Once an archive is unmounted, its contents will no longer be available at the mount point.

Notes

A dearchiver must be present in the I/O filter stack for this function to succeed.

See Also

sceFiosIOFilterPsarcDearchiver(), sceFiosArchiveMount()

sceFiosArchiveUnmountSync

Unmounts a previously mounted archive (sync).

Definition

```
#include <fios2_api.h>
int sceFiosArchiveUnmountSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh
);
```

Arguments

[in] *pAttr* Operation attributes. May be NULL.
[in] *fh* File handle to the archive.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Unmounts a previously mounted archive (sync). Once an archive is unmounted, its contents will no longer be available at the mount point.

Notes

A dearchiver must be present in the I/O filter stack for this function to succeed.

See Also

sceFiosIOFilterPsarcDearchiver(), sceFiosArchiveMountSync()

sceFiosCacheContainsFileRangeSync

Indicates whether FIOS's data caches contain part of a given file.

Definition

```
#include <fios2_api.h>
int sceFiosCacheContainsFileRangeSync(
    const char *pPath,
    SceFiosOffset offset,
    SceFiosSize length
);
```

Arguments

[in] *pPath*	Path to file to query.
[in] *offset*	Offset within file to query.
[in] *length*	Number of bytes within file to query.

Return Values

Returns SCE_FIOS_ERROR_BAD_INDEX if any of the range is not cached. Otherwise, it returns the filter index of the cache containing the data.

Description

Because of the dynamic nature of the cache, the answer could change as more I/O is issued. It is possible that this function could return true and yet the data could be evicted by the time you try to read it, or vice-versa.

sceFiosCacheContainsFileSync

Indicates whether FIOS's data caches contain all of a given file.

Definition

```
#include <fios2_api.h>
int sceFiosCacheContainsFileSync(
    const char *pPath,
    SceFiosSize *pOutSize
);
```

Arguments

[in] *pPath*	Path to file to query.
[out] *pOutSize*	On success, filled with the file's size. May be NULL.

Return Values

Returns SCE_FIOS_ERROR_BAD_INDEX if any of the file is not cached. Otherwise, it returns the filter index of the cache containing the data.

Description

Indicates whether FIOS's data caches contain all of a given file.

Notes

This function is synchronous and will block while it stats the file to get the file size.

sceFiosCacheFlushFileRangeSync

Flushes part of a specified file from FIOS's data caches.

Definition

```
#include <fios2_api.h>
void sceFiosCacheFlushFileRangeSync(
    const char *pPath,
    SceFiosOffset offset,
    SceFiosSize length
);
```

Arguments

[in] *pPath*	File to flush.
[in] *offset*	Offset within the file to flush.
[in] *length*	Number of bytes from the offset to flush.

Return Values

None

Description

This call can be used when you know (from an external source) that part of a file has changed and it needs to be reloaded from the media.

sceFiosCacheFlushFileSync

Flushes a specified file from FIOS's data caches.

Definition

```
#include <fios2_api.h>
void sceFiosCacheFlushFileSync(
    const char *pPath
);
```

Arguments

[in] *pPath* File to flush.

Return Values

None

Description

This call can be used when you know (from an external source) that a file has changed and it needs to be reloaded from the media.

sceFiosCancelAllOps

Cancels any outstanding operations.

Definition

```
#include <fios2_api.h>
void sceFiosCancelAllOps();
```

Arguments

None

Return Values

None

Description

Cancels any outstanding operations. This cancels all operations except for close operations like sceFiosFHClose() or sceFiosDHClose().

sceFiosChangeStat

Changes the status of a file or directory.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosChangeStat(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    SceFiosStat *pStatus,
    int cbit
);
```

Arguments

[in] *pAttr*	Operation attributes. May be `NULL`.
[in] *pPath*	Path to modify.
[in] *pStatus*	A `SceFiosStat` structure that contains the values to change.
[in] *cbit*	A bit pattern that specifies which values to change. Can contain any combination of: – `SCE_FIOS_CST_MODE` Change read/write permissions. – `SCE_FIOS_CST_SIZE` Change file size. – `SCE_FIOS_CST_CT` Change creation date/time. – `SCE_FIOS_CST_AT` Change last access data/time. – `SCE_FIOS_CST_MT` Change last modification date/time.

Return Values

`SCE_FIOS_OK` for success, or an error code.

Description

Changes the status of a file or directory. Available only on PlayStation®Vita.

See Also

`sceFiosStat()`

sceFiosChangeStatSync

Changes the status of a file or directory (sync).

Definition

```
#include <fios2_api.h>
int sceFiosChangeStatSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    SceFiosStat *pStatus,
    int cbit
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to modify.
[in] *pStatus*	A SceFiosStat structure that contains the values to change.
[in] *cbit*	A bit pattern that specifies which values to change.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Changes the status of a file or directory (sync). Available only on PlayStation®Vita.

Notes

On PlayStation®Vita, if the statFlags member of the SceFiosStat structure is 0 it is assumed that the *mode* member has platform-specific values to use for file permissions. See SceIoMode in the *Kernel Reference*.

See Also

sceFiosStat()

sceFiosCloseAllFiles

Closes any open file handles.

Definition

```
#include <fios2_api.h>
void sceFiosCloseAllFiles();
```

Arguments

None

Return Values

None

Description

Closes any open file handles.

sceFiosDateFromComponents

Converts a `struct tm` into a `SceFiosDate`.

Definition

```
#include <fios2_api.h>
SceFiosDate sceFiosDateFromComponents(
    const struct tm *pComponents
);
```

Arguments

[in] *pComponents* Pointer to a `struct tm`.

Return Values

An equivalent date, or 0 if the date could not be represented.

Description

Converts a `struct tm` into a `SceFiosDate`.

Notes

The *tm_wday*, *tm_yday* and *tm_isdst* fields in *pComponents* are ignored. In addition, the original values of the other components are not restricted to their normal ranges, and are normalized as needed. Examples: October 40 is changed to November 9, a *tm_hour* of -1 means 1 hour before midnight, *tm_mday* of 0 means the day preceding the current month, and *tm_mon* of -2 means 2 months before January of *tm_year*. See `time.h` for a description of `struct tm`.

sceFiosDateFromFILETIME

Converts a `FILETIME` into a `SceFiosDate`.

Definition

```
#include <fios2_api.h>
SceFiosDate sceFiosDateFromFILETIME(
        const FILETIME *ft
);
```

Arguments

[in] *ft* `FILETIME` to convert.

Return Values

The equivalent `SceFiosDate`, or 0 if the date could not be represented.

Description

Converts a `FILETIME` into a `SceFiosDate`. Available only on Windows.

Notes

For information about `FILETIME`, see `WinDef.h` (in `\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Include`).

sceFiosDateFromSceDateTime

Converts a `SceDateTime` (see the *libscebase Reference*) into a `SceFiosDate`.

Definition

```
#include <fios2_api.h>
SceFiosDate sceFiosDateFromSceDateTime(
    const SceDateTime *pSceDateTime
);
```

Arguments

[in] *pSceDateTime* Pointer to a `SceDateTime`.

Return Values

The equivalent `SceFiosDate`, or 0 if the date could not be represented.

Description

Converts a `SceDateTime` (see the *libscebase Reference*) into a `SceFiosDate`. Available only on PlayStation®Vita.

sceFiosDateGetCurrent

Gets the current date.

Definition

```
#include <fios2_api.h>
SceFiosDate sceFiosDateGetCurrent();
```

Arguments

None

Return Values

The current date.

Description

Gets the current date.

sceFiosDateToComponents

Converts a SceFiosDate into a `struct tm`.

Definition

```
#include <fios2_api.h>
struct tm *sceFiosDateToComponents(
    SceFiosDate date,
    struct tm *pOutComponents
);
```

Arguments

[in] *date*	Date to convert.
[out] *pOutComponents*	Filled in with the date components.

Return Values

The result is always equal to *pOutComponents*.

Description

Converts a SceFiosDate into a `struct tm`.

Notes

The *tm_wday*, *tm_yday* and *tm_isdst* fields in *pOutComponents* are not updated. See `time.h` for a description of `struct tm`.

sceFiosDateToSceDateTime

Converts a SceFiosDate into a SceDateTime (see the *libscebase Reference*).

Definition

```
#include <fios2_api.h>
SceDateTime *sceFiosDateToSceDateTime(
    SceFiosDate date,
    SceDateTime *pSceDateTime
);
```

Arguments

[in] *date*	SceFiosDate to convert.
[out] *pSceDateTime*	Filled with the result for the converted date.

Return Values

The result is always equal to *pSceDateTime*.

Description

Converts a SceFiosDate into a SceDateTime (see the *libscebase Reference*). Available only on PlayStation®Vita.

sceFiosDebugDumpDate

Creates a human-readable debug description of a `SceFiosDate`.

Definition

```
#include <fios2_debug.h>
char *sceFiosDebugDumpDate(
    SceFiosDate date,
    char *pBuffer,
    size_t bufferSize
);
```

Arguments

[in] *date*	Date to dump.
[in] *pBuffer*	Buffer to receive the dump. May be `NULL`.
[in] *bufferSize*	Size of the buffer. For best results, the buffer should be at least 48 characters.

Return Values

If *pBuffer* was non-`NULL`, it is returned. Otherwise an internal global buffer is returned.

Description

Creates a human-readable debug description of a `SceFiosDate`.

sceFiosDebugDumpDH

Creates a human-readable debug description showing the details of a directory handle.

Definition

```
#include <fios2_debug.h>
char *sceFiosDebugDumpDH(
    SceFiosDH dh,
    char *pBuffer,
    size_t bufferSize
);
```

Arguments

[in] *dh*	Directory handle to dump.
[in] *pBuffer*	Buffer to receive the dump. May be NULL.
[in] *bufferSize*	Size of the buffer. For best results, the buffer should be least 512 characters.

Return Values

If *pBuffer* was non-NULL, it is returned. Otherwise an internal global buffer is returned.

Description

Creates a human-readable debug description showing the details of a directory handle.

sceFiosDebugDumpError

Creates a human-readable debug description of an error returned by FIOS.

Definition

```
#include <fios2_debug.h>
char *sceFiosDebugDumpError(
    int err,
    char *pBuffer,
    size_t bufferSize
);
```

Arguments

[in] *err*	Error code.
[in] *pBuffer*	Buffer to receive the error string. May be NULL.
[in] *bufferSize*	Size of the buffer. For best results, the buffer should be least 48 characters.

Return Values

If *pBuffer* was non-NULL, it is returned. Otherwise an internal global buffer is returned.

Description

Creates a human-readable debug description of an error returned by FIOS.

sceFiosDebugDumpFH

Creates a human-readable debug description showing the details of a file handle.

Definition

```
#include <fios2_debug.h>
char *sceFiosDebugDumpFH(
    SceFiosFH fh,
    char *pBuffer,
    size_t bufferSize
);
```

Arguments

[in] *fh*	File handle to dump.
[in] *pBuffer*	Buffer to receive the dump. May be NULL.
[in] *bufferSize*	Size of the buffer. For best results, the buffer should be least 512 characters.

Return Values

If *pBuffer* was non-NULL, it is returned. Otherwise an internal global buffer is returned.

Description

Creates a human-readable debug description showing the details of a file handle.

sceFiosDebugDumpOp

Creates a human-readable debug description showing the details of an op handle.

Definition

```
#include <fios2_debug.h>
char *sceFiosDebugDumpOp(
    SceFiosOp op,
    char *pBuffer,
    size_t bufferSize
);
```

Arguments

[in] *op*	Operation to dump.
[in] *pBuffer*	Buffer to receive the dump. May be NULL.
[in] *bufferSize*	Size of the buffer. For best results, the buffer should be least 512 characters.

Return Values

If *pBuffer* was non-NULL, it is returned. Otherwise an internal global buffer is returned.

Description

Creates a human-readable debug description showing the details of an op handle.

sceFiosDelete

Deletes a file or directory.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosDelete(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	File or directory to delete.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Deletes a file or directory.

Notes

This function will fail if used on a non-empty directory.

sceFiosDeleteSync

Deletes a file or directory (sync).

Definition

```
#include <fios2_api.h>
int sceFiosDeleteSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr* Operation attributes. May be NULL.
[in] *pPath* File or directory to delete.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Deletes a file or directory (sync).

Notes

This function will fail if used on a non-empty directory.

sceFiosDevctl

Performs a device-specific control operation.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosDevctl(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    int cmd,
    void *pArg,
    size_t arglen,
    void *pBuffer,
    size_t buflen
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to the device.
[in] *cmd*	The device-specific command code.
[in] *pArg*	Pointer to a device-dependent parameter block.
[in] *arglen*	Size of the device-dependent parameter block.
[out] *pBuffer*	Pointer to a return-data storage block.
[in] *buflen*	Size of the return-data storage block.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Performs a device-specific control operation. Available only on PlayStation®Vita.

Notes

This operation, along with its associated data, is device- and platform-specific.

sceFiosDevctlSync

Performs a device-specific control operation (sync).

Definition

```
#include <fios2_api.h>
int sceFiosDevctlSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    int cmd,
    void *pArg,
    size_t arglen,
    void *pBuffer,
    size_t buflen
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to the device.
[in] *cmd*	The device-specific command code.
[in] *pArg*	Pointer to a device-dependent parameter block.
[in] *arglen*	Size of the device-dependent parameter block.
[out] *pBuffer*	Pointer to a return-data storage block.
[in] *buflen*	Size of the return-data storage block.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Performs a device-specific control operation (sync). Available only on PlayStation®Vita.

Notes

This operation, along with its associated data, is device- and platform-specific.

sceFiosDHClose

Closes a directory.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosDHClose(
    const SceFiosOpAttr *pAttr,
    SceFiosDH dh
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *dh*	Directory handle.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Closes a directory.

See Also

sceFiosDHOpen(), sceFiosDHOpenSync(), sceFiosDHCloseSync()

sceFiosDHCloseSync

Closes a directory (sync).

Definition

```
#include <fios2_api.h>
int sceFiosDHCloseSync(
    const SceFiosOpAttr *pAttr,
    SceFiosDH dh
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *dh*	Directory handle.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Closes a directory (sync).

See Also

sceFiosDHOpen(), sceFiosDHOpenSync(), sceFiosDHClose()

sceFiosDHGetPath

Returns the path of an open directory handle.

Definition

```
#include <fios2_api.h>
const char *sceFiosDHGetPath(
    SceFiosDH dh
);
```

Arguments

[in] *dh* Directory handle.

Return Values

Path, or NULL if the directory handle was invalid.

Description

Returns the path of an open directory handle.

See Also

sceFiosDHOpen()

sceFiosDHOpen

Opens a directory.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosDHOpen(
    const SceFiosOpAttr *pAttr,
    SceFiosDH *pOutDH,
    const char *pPath,
    SceFiosBuffer buf
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[out] *pOutDH*	Returns a directory handle, which can be used immediately – even before the open has completed.
[in] *pPath*	Path to a directory to open.
[in] *buf*	Buffer to use when reading directory entries. No buffer is required on PlayStation®Vita. On other platforms, a buffer might be required. The size of the buffer can be determined by passing a buffer with length zero, then using sceFiosOpWait() to wait for the op to complete and then using sceFiosOpGetActualCount() to query the size (for performance reasons, the size returned is not the absolute minimum required size). If a buffer of inadequate length is passed in, the op will fail with SCE_FIOS_ERROR_BAD_SIZE. If the buffer's length is larger than the maximum allowed size, the value returned by sceFiosOpGetActualCount() after SCE_FIOS_ERROR_BAD_SIZE will be -1.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Opens a directory.

Notes

When working with non-opaque overlays, this function supports at most the following number of directory entries:

- For Win64, the maximum is 5127 entries.
- For Win32, the maximum is 5191 entries.
- For PlayStation®Vita, the maximum is 282 entries.

See Also

sceFiosDHOpenSync(), sceFiosDHRead(), sceFiosDHReadSync(), sceFiosDHClose(), sceFiosDHCloseSync(), sceFiosDHGetPath()

sceFiosDHOpenSync

Opens a directory (sync).

Definition

```
#include <fios2_api.h>
int sceFiosDHOpenSync(
    const SceFiosOpAttr *pAttr,
    SceFiosDH *pOutDH,
    const char *pPath,
    SceFiosBuffer buf
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[out] *pOutDH*	Returns a directory handle.
[in] *pPath*	Path to a directory to open.
[in] *buf*	Buffer to use when reading directory entries. No buffer is required on PlayStation®Vita. On other platforms, a buffer might be required. If a buffer is required and a buffer of inadequate length is passed in, the op will fail with SCE_FIOS_ERROR_BAD_SIZE. The size of the buffer can be determined by passing a buffer of length zero to the async sceFiosDHOpen() function. See its documentation for details.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Opens a directory (sync).

Notes

When working with non-opaque overlays, this function supports at most the following number of directory entries:

- For Win64, the maximum is 5127 entries.
- For Win32, the maximum is 5191 entries.
- For PlayStation®Vita, the maximum is 282 entries.

See Also

sceFiosDHOpen(), sceFiosDHRead(), sceFiosDHReadSync(), sceFiosDHClose(), sceFiosDHCloseSync(), sceFiosDHGetPath()

sceFiosDHRead

Reads an open directory.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosDHRead(
    const SceFiosOpAttr *pAttr,
    SceFiosDH dh,
    SceFiosDirEntry *pOutEntry
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *dh*	Directory handle.
[out] *pOutEntry*	Upon successful completion, filled with a SceFiosDirEntry structure.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Reads an open directory.

See Also

sceFiosDHOpen(), sceFiosDHOpenSync(), sceFiosDHReadSync(), sceFiosDHClose(), sceFiosDHCloseSync()

sceFiosDHReadSync

Reads an open directory (sync).

Definition

```
#include <fios2_api.h>
int sceFiosDHReadSync(
    const SceFiosOpAttr *pAttr,
    SceFiosDH dh,
    SceFiosDirEntry *pOutEntry
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *dh*	Directory handle.
[out] *pOutEntry*	Upon successful completion, filled with a SceFiosDirEntry structure.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Reads an open directory (sync).

See Also

sceFiosDHOpen(), sceFiosDHOpenSync(), sceFiosDHRead(), sceFiosDHClose(), sceFiosDHCloseSync()

sceFiosDirectoryCreate

Creates a directory.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosDirectoryCreate(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to the new directory to create.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Creates a directory.

Notes

Intermediate directories are not created. All components of the path except for the final component must already exist.

sceFiosDirectoryCreateSync

Creates a directory (sync).

Definition

```
#include <fios2_api.h>
int sceFiosDirectoryCreateSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr* Operation attributes. May be NULL.
[in] *pPath* Path to the new directory to create.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Creates a directory (sync).

Notes

Intermediate directories are not created. All components of the path except for the final component must already exist.

sceFiosDirectoryCreateWithMode

Creates a directory.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosDirectoryCreateWithMode(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    int32_t nativeMode
);
```

Arguments

[in] *pAttr*	Operation attributes. May be `NULL`.
[in] *pPath*	Path to the new directory to create.
[in] *nativeMode*	OS-specific mode (permissions) flag. For PlayStation®Vita, see `SceIoMode` in the *Kernel Reference*.

Return Values

Operation handle. Caller is responsible for deleting this handle with `sceFiosOpDelete()`.

Description

Creates a directory.

Notes

Intermediate directories are not created. All components of the path except for the final component must already exist.

sceFiosDirectoryCreateWithModeSync

Creates a directory (sync).

Definition

```
#include <fios2_api.h>
int sceFiosDirectoryCreateWithModeSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    int32_t nativeMode
);
```

Arguments

[in] *pAttr*	Operation attributes. May be `NULL`.
[in] *pPath*	Path to the new directory to create.
[in] *nativeMode*	OS-specific mode (permissions) flag. For PlayStation®Vita, see `SceIoMode` in the *Kernel Reference*.

Return Values

`SCE_FIOS_OK` for success, or an error code.

Description

Creates a directory (sync).

Notes

Intermediate directories are not created. All components of the path except for the final component must already exist.

sceFiosDirectoryDelete

Deletes a directory.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosDirectoryDelete(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Directory to delete.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Deletes a directory.

Notes

This function will fail if used on a file or on a non-empty directory.

sceFiosDirectoryDeleteSync

Deletes a directory (sync).

Definition

```
#include <fios2_api.h>
int sceFiosDirectoryDeleteSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Directory to delete.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Deletes a directory (sync).

Notes

This function will fail if used on a file or on a non-empty directory.

sceFiosDirectoryExists

Indicates whether a directory exists at a path.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosDirectoryExists(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    bool *pOutExists
);
```

Arguments

[in] *pAttr*	Operation attributes. May be `NULL`.
[in] *pPath*	Path to query.
[out] *pOutExists*	Upon successful completion, set to `true` or `false`.

Return Values

Operation handle. Caller is responsible for deleting this handle with `sceFiosOpDelete()`.

Description

Indicates whether a directory exists at a path.

Notes

This function returns `false` if a file exists at the path instead of a directory.

sceFiosDirectoryExistsSync

Indicates whether a directory exists at a path (sync).

Definition

```
#include <fios2_api.h>
bool sceFiosDirectoryExistsSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be `NULL`.
[in] *pPath*	Path to query.

Return Values

Returns `true` if a directory exists at the path, `false` if any error occurred.

Description

Indicates whether a directory exists at a path (sync).

Notes

This function returns `false` if a file exists at the path instead of a directory.

sceFiosExists

Indicates whether any item exists at a path.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosExists(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    bool *pOutExists
);
```

Arguments

[in] *pAttr*	Operation attributes. May be `NULL`.
[in] *pPath*	Path to query.
[out] *pOutExists*	Upon successful completion, set to `true` or `false`.

Return Values

Operation handle. Caller is responsible for deleting this handle with `sceFiosOpDelete()`.

Description

Indicates whether any item exists at a path.

Notes

This function will return `true` if either a file or a directory exists at the path.

sceFiosExistsSync

Indicates whether any item exists at a path (sync).

Definition

```
#include <fios2_api.h>
bool sceFiosExistsSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be `NULL`.
[in] *pPath*	Path to query.

Return Values

Returns `true` if any item exists at the path, `false` if any error occurred.

Description

Indicates whether any item exists at a path (sync).

Notes

This function returns `true` if either a file or a directory exists at the path.

sceFiosFHClose

Closes an open file.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHClose(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Closes an open file.

See Also

sceFiosFHOpen()

sceFiosFHCloseSync

Closes an open file (sync).

Definition

```
#include <fios2_api.h>
int sceFiosFHCloseSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Closes an open file (sync).

See Also

sceFiosFHOpenSync()

sceFiosFHGetOpenParams

Returns the parameters used to open a file.

Definition

```
#include <fios2_api.h>
SceFiosOpenParams *sceFiosFHGetOpenParams(
    SceFiosFH fh
);
```

Arguments

[in] *fh* File handle.

Return Values

Open parameters, or `NULL` if the file handle is invalid.

Description

Returns the parameters used to open a file.

See Also

sceFiosFHOpen(), sceFiosFHOpenSync()

Document serial number: 000004892117

sceFiosFHGetPath

Returns the path of an open file handle.

Definition

```
#include <fios2_api.h>
const char *sceFiosFHGetPath(
    SceFiosFH fh
);
```

Arguments

[in] *fh* File handle.

Return Values

Path, or NULL if the file handle is invalid.

Description

Returns the path of an open file handle.

sceFiosFHGetSize

Returns the size of an open file.

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFHGetSize(
    SceFiosFH fh
);
```

Arguments

[in] *fh* File handle.

Return Values

Size, or `-1` if the file handle is invalid.

Description

Returns the size of an open file.

sceFiosFHIoctl

Performs a device-specific or file-specific control operation.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHIoctl(
    const SceFiosOpAttr *pAttr,
    const SceFiosFH fh,
    int cmd,
    void *pArg,
    size_t arglen,
    void *pBuffer,
    size_t buflen
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle for the device or file.
[in] *cmd*	The device-specific command code.
[in] *pArg*	Pointer to a device-dependent parameter block.
[in] *arglen*	Size of the device-dependent parameter block.
[out] *pBuffer*	Pointer to a return-data storage block.
[in] *buflen*	Size of the return-data storage block.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Performs a device-specific or file-specific control operation. Available only on PlayStation®Vita.

Notes

This operation, along with its associated data, is device- and platform-specific.

sceFiosFHIoctlSync

Performs a device-specific or file-specific control operation (sync).

Definition

```
#include <fios2_api.h>
int sceFiosFHIoctlSync(
    const SceFiosOpAttr *pAttr,
    const SceFiosFH fh,
    int cmd,
    void *pArg,
    size_t arglen,
    void *pBuffer,
    size_t buflen
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle for the device or file.
[in] *cmd*	The device-specific command code.
[in] *pArg*	Pointer to a device-dependent parameter block.
[in] *arglen*	Size of the device-dependent parameter block.
[out] *pBuffer*	Pointer to a return-data storage block.
[in] *buflen*	Size of the return-data storage block.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Performs a device-specific or file-specific control operation (sync). Available only on PlayStation®Vita.

Notes

This operation, along with its associated data, is device- and platform-specific.

sceFiosFHOpen

Opens a file.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHOpen(
    const SceFiosOpAttr *pAttr,
    SceFiosFH *pOutFH,
    const char *pPath,
    const SceFiosOpenParams *pOpenParams
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[out] *pOutFH*	Returns a file handle, which can be used immediately – even before the open has completed.
[in] *pPath*	Path to a file to open.
[in] *pOpenParams*	Open parameters. May be NULL, in which case the file will be opened with normal read-only permission.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Opens a file.

See Also

sceFiosFHClose()

sceFiosFHOpenSync

Opens a file (sync).

Definition

```
#include <fios2_api.h>
int sceFiosFHOpenSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH *pOutFH,
    const char *pPath,
    const SceFiosOpenParams *pOpenParams
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[out] *pOutFH*	Returns a file handle.
[in] *pPath*	Path to a file to open.
[in] *pOpenParams*	Open parameters. May be NULL, in which case the file will be opened with normal read-only permission.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Opens a file (sync).

See Also

sceFiosFHCloseSync()

sceFiosFHOpenWithMode

Opens a file with creation mode.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHOpenWithMode(
    const SceFiosOpAttr *pAttr,
    SceFiosFH *pOutFH,
    const char *pPath,
    const SceFiosOpenParams *pOpenParams,
    int32_t nativeMode
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[out] *pOutFH*	Returns a file handle, which can be used immediately – even before the open has completed.
[in] *pPath*	Path to a file to open.
[in] *pOpenParams*	Open parameters. May be NULL, in which case the file will be opened with normal read-only permission.
[in] *nativeMode*	OS-specific mode (permissions) flag, only used if SCE_FIOS_O_CREAT is specified. For PlayStation®Vita, see SceIoMode in the *Kernel Reference*.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Opens a file with creation mode.

See Also

sceFiosFHClose()

sceFiosFHOpenWithModeSync

Opens a file with creation mode (sync).

Definition

```
#include <fios2_api.h>
int sceFiosFHOpenWithModeSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH *pOutFH,
    const char *pPath,
    const SceFiosOpenParams *pOpenParams,
    int32_t nativeMode
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[out] *pOutFH*	Returns a file handle.
[in] *pPath*	Path to a file to open.
[in] *pOpenParams*	Open parameters. May be NULL, in which case the file will be opened with normal read-only permission.
[in] *nativeMode*	OS-specific mode (permissions) flag, only used if SCE_FIOS_O_CREAT is specified. For PlayStation®Vita, see SceIoMode in the *Kernel Reference*.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Opens a file with creation mode (sync).

See Also

sceFiosFHCloseSync()

sceFiosFHPread

Reads bytes from a file, at a specified offset.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHPread(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    void *pBuf,
    SceFiosSize length,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *pBuf*	Buffer to receive file data.
[in] *length*	Requested number of bytes to read.
[in] *offset*	Offset to read from.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Reads bytes from a file, at a specified offset.

Notes

This call does not use or modify the file handle's internal file position.

sceFiosFHPreadSync

Reads bytes from a file, at a specified offset (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFHPreadSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    void *pBuf,
    SceFiosSize length,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *pBuf*	Buffer to receive file data.
[in] *length*	Requested number of bytes to read.
[in] *offset*	Offset to read from.

Return Values

Non-negative values are the number of bytes read; 0 <= result <= length. Negative values are error codes. Return of zero indicates a read offset starting at or after the end-of-file.

Description

Reads bytes from a file, at a specified offset (sync).

Notes

This call does not use or modify the file handle's internal file position.

sceFiosFHPreadv

Reads bytes from a file, at a specified offset, to multiple destination buffers.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHPreadv(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const SceFiosBuffer iov[],
    int iovcnt,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *iov*	Array of buffer descriptors to receive the file data.
[in] *iovcnt*	Size of the *iov* array. The maximum size for this array is 8.
[in] *offset*	Offset to read from.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Reads bytes from a file, at a specified offset, to multiple destination buffers.

Notes

This call does not use or modify the file handle's internal file position.

sceFiosFHPreadvSync

Reads bytes from a file, at a specified offset, to multiple destination buffers (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFHPreadvSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const SceFiosBuffer iov[],
    int iovcnt,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *iov*	Array of buffer descriptors to receive file data.
[in] *iovcnt*	Size of the *iov* array. The maximum size for this array is 8.
[in] *offset*	Offset to read from.

Return Values

Non-negative values are the number of bytes read; 0 <= result <= (sum of iov length values). Negative values are error codes. Return of zero indicates a read offset starting at or after the end-of-file.

Description

Reads bytes from a file, at a specified offset, to multiple destination buffers (sync).

Notes

This call does not use or modify the file handle's internal file position.

sceFiosFHPwrite

Writes bytes to a file, at a specified offset.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHPwrite(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const void *pBuf,
    SceFiosSize length,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *pBuf*	Buffer with file data to write.
[in] *length*	Requested number of bytes to write.
[in] *offset*	Offset to write to.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Writes bytes to a file, at a specified offset.

Notes

This call does not use or modify the file handle's internal file position.

sceFiosFHPwriteSync

Writes bytes to a file, at a specified offset (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFHPwriteSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const void *pBuf,
    SceFiosSize length,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *pBuf*	Buffer with file data to write.
[in] *length*	Requested number of bytes to write.
[in] *offset*	Offset to write to.

Return Values

Non-negative values are the number of bytes written; 0 <= result <= length. Negative values are error codes.

Description

Writes bytes to a file, at a specified offset (sync).

Notes

This call does not use or modify the file handle's internal file position.

sceFiosFHPwritev

Writes bytes to a file, at a specified offset, with multiple source buffers.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHPwritev(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const SceFiosBuffer iov[],
    int iovcnt,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *iov*	Array of buffer descriptors containing file data to write.
[in] *iovcnt*	Size of the *iov* array. The maximum size for this array is 8.
[in] *offset*	Offset to write to.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Writes bytes to a file, at a specified offset, with multiple source buffers.

Notes

This call does not use or modify the file handle's internal file position.

sceFiosFHPwritevSync

Writes bytes to a file, at a specified offset, with multiple source buffers (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFHPwritevSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const SceFiosBuffer iov[],
    int iovcnt,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *iov*	Array of buffer descriptors containing file data to write.
[in] *iovcnt*	Size of the *iov* array. The maximum size for this array is 8.
[in] *offset*	Offset to write to.

Return Values

Non-negative values are the number of bytes written; 0 <= result <= (sum of iov length values).
Negative values are error codes.

Description

Writes bytes to a file, at a specified offset, with multiple source buffers (sync).

Notes

This call does not use or modify the file handle's internal file position.

sceFiosFHRead

Reads bytes from a file.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHRead(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    void *pBuf,
    SceFiosSize length
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *pBuf*	Buffer to receive file data.
[in] *length*	Requested number of bytes to read.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Reads bytes from a file.

Notes

This call uses and modifies the file handle's internal file position, but does not update it until after the operation completes. Thus, issuing multiple calls to this function at the same time for the same file handle could lead to unexpected results; to avoid this problem, use sceFiosFHPread().

sceFiosFHReadSync

Reads bytes from a file (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFHReadSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    void *pBuf,
    SceFiosSize length
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *pBuf*	Buffer to receive file data.
[in] *length*	Requested number of bytes to read.

Return Values

Non-negative values are the number of bytes read; 0 <= result <= length. Negative values are error codes. Return of zero indicates a read offset starting at or after the end-of-file.

Description

Reads bytes from a file (sync).

Notes

This call uses and modifies the file handle's internal file position, but does not update it until after the operation completes. Thus, issuing multiple calls to this function from multiple threads at the same time for the same file handle could lead to unexpected results; to avoid this problem, use sceFiosFHPreadSync().

sceFiosFHReadv

Reads bytes from a file, with multiple destination buffers.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHReadv(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const SceFiosBuffer iov[],
    int iovcnt
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *iov*	Array of buffer descriptors to receive the file data.
[in] *iovcnt*	Size of the *iov* array. The maximum size for this array is 8.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Reads bytes from a file, with multiple destination buffers.

Notes

This call uses and modifies the file handle's internal file position.

sceFiosFHReadvSync

Reads bytes from a file, with multiple destination buffers (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFHReadvSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const SceFiosBuffer iov[],
    int iovcnt
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *iov*	Array of buffer descriptors to receive file data.
[in] *iovcnt*	Size of the *iov* array. The maximum size for this array is 8.

Return Values

Non-negative values are the number of bytes read; 0 <= result <= (sum of iov length values). Negative values are error codes. Return of zero indicates a read offset starting at or after the end-of-file.

Description

Reads bytes from a file, with multiple destination buffers (sync).

Notes

This call uses and modifies the file handle's internal file position.

sceFiosFHSeek

Seeks within an open file.

Definition

```
#include <fios2_api.h>
SceFiosOffset sceFiosFHSeek(
    SceFiosFH fh,
    SceFiosOffset offset,
    SceFiosWhence whence
);
```

Arguments

[in] *fh*	File handle.
[in] *offset*	Offset to seek to.
[in] *whence*	How to seek (from start, current location, or end).

Return Values

New file position, as a number of bytes from the start of the file, or `-1` if the file handle is invalid.

Description

Seeks within an open file.

See Also

`sceFiosFHTell()`

sceFiosFHStat

Returns a full set of status information for an open file.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHStat(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    SceFiosStat *pOutStatus
);
```

Arguments

[in] *pAttr*	Operation attributes. May be `NULL`.
[in] *fh*	File handle.
[out] *pOutStatus*	Upon successful completion, filled in with a `SceFiosStat` structure.

Return Values

Operation handle. Caller is responsible for deleting this handle with `sceFiosOpDelete()`.

Description

Returns a full set of status information for an open file.

Document serial number: 000004892117

sceFiosFHStatSync

Returns a full set of status information for an open file (sync).

Definition

```
#include <fios2_api.h>
int sceFiosFHStatSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    SceFiosStat *pOutStatus
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[out] *pOutStatus*	Upon successful completion, filled in with a SceFiosStat structure.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Returns a full set of status information for an open file (sync).

sceFiosFHSync

Flushes changes to a file that is open for writing.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Flushes changes to a file that is open for writing. This function might fail if the underlying media doesn't support the sync command. For example, sceFiosFHSync() might return an error for files on the network.

sceFiosFHSyncSync

Flushes changes to a file that is open for writing (sync).

Definition

```
#include <fios2_api.h>
int sceFiosFHSyncSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Flushes changes to a file that is open for writing (sync). This function might fail if the underlying media doesn't support the sync command. For example, on some platforms sceFiosFHSyncSync() will return an error for files on HOSTFS.

sceFiosFHTell

Returns the current file position of an open file.

Definition

```
#include <fios2_api.h>
SceFiosOffset sceFiosFHTell(
    SceFiosFH fh
);
```

Arguments

[in] *fh* File handle.

Return Values

Current file position, or `-1` if the file handle is invalid.

Description

Returns the current file position of an open file.

See Also

`sceFiosFHSeek()`

sceFiosFHTruncate

Resizes an open file.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHTruncate(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    SceFiosSize length
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *length*	New file size.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Resizes an open file.

sceFiosFHTruncateSync

Resizes an open file (sync).

Definition

```
#include <fios2_api.h>
int sceFiosFHTruncateSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    SceFiosSize length
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *length*	New file size.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Resizes an open file (sync).

sceFiosFHWrite

Writes bytes to a file.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHWrite(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const void *pBuf,
    SceFiosSize length
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *pBuf*	Buffer with file data to write.
[in] *length*	Requested number of bytes to write.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Writes bytes to a file.

Notes

This call uses and modifies the file handle's internal file position.

sceFiosFHWriteSync

Writes bytes to a file (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFHWriteSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const void *pBuf,
    SceFiosSize length
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *pBuf*	Buffer with file data to write.
[in] *length*	Requested number of bytes to write.

Return Values

Non-negative values are the number of bytes written; 0 <= result <= length. Negative values are error codes.

Description

Writes bytes to a file (sync).

Notes

This call uses and modifies the file handle's internal file position.

sceFiosFHWritev

Writes bytes to a file, with multiple source buffers.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFHWritev(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const SceFiosBuffer iov[],
    int iovcnt
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *iov*	Array of buffer descriptors containing file data to write.
[in] *iovcnt*	Size of the *iov* array. The maximum size for this array is 8.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Writes bytes to a file, with multiple source buffers.

Notes

This call uses and modifies the file handle's internal file position.

sceFiosFHWritevSync

Writes bytes to a file, with multiple source buffers (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFHWritevSync(
    const SceFiosOpAttr *pAttr,
    SceFiosFH fh,
    const SceFiosBuffer iov[],
    int iovcnt
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *fh*	File handle.
[in] *iov*	Array of buffer descriptors containing file data to write.
[in] *iovcnt*	Size of the *iov* array. The maximum size for this array is 8.

Return Values

Non-negative values are the number of bytes written; 0 <= result <= (sum of iov length values).
Negative values are error codes.

Description

Writes bytes to a file, with multiple source buffers (sync).

Notes

This call uses and modifies the file handle's internal file position.

sceFiosFileDelete

Deletes a file.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFileDelete(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	File to delete.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Deletes a file.

Notes

This function will fail if used on a directory.

sceFiosFileDeleteSync

Deletes a file (sync).

Definition

```
#include <fios2_api.h>
int sceFiosFileDeleteSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	File to delete.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Deletes a file (sync).

Notes

This function will fail if used on a directory.

sceFiosFileExists

Indicates whether a file exists at a path.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFileExists(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    bool *pOutExists
);
```

Arguments

[in] *pAttr*	Operation attributes. May be `NULL`.
[in] *pPath*	Path to query.
[out] *pOutExists*	Upon successful completion, set to `true` or `false`.

Return Values

Operation handle. Caller is responsible for deleting this handle with `sceFiosOpDelete()`.

Description

Indicates whether a file exists at a path.

Notes

This function returns `false` if a directory exists at the path instead of a file.

sceFiosFileExistsSync

Indicates whether a file exists at a path (sync).

Definition

```
#include <fios2_api.h>
bool sceFiosFileExistsSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be `NULL`.
[in] *pPath*	Path to query.

Return Values

Returns `true` if a file exists at the path, `false` if any error occurred.

Description

Indicates whether a file exists at a path (sync).

Notes

This function returns `false` if a directory exists at the path instead of a file.

sceFiosFileGetSize

Returns the size of a file.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFileGetSize(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    SceFiosSize *pOutSize
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to query.
[out] *pOutSize*	File size, or a negative error code.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Returns the size of a file.

sceFiosFileGetSizeSync

Returns the size of a file (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFileGetSizeSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to query.

Return Values

File size, or a negative error code.

Description

Returns the size of a file (sync).

sceFiosFileRead

Reads bytes from a file without opening it first.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFileRead(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    void *pBuf,
    SceFiosSize length,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to a file to read.
[in] *pBuf*	Buffer to receive file data.
[in] *length*	Requested number of bytes to read.
[in] *offset*	Offset within the file to read from.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Reads bytes from a file without opening it first. This call is equivalent to sceFiosFHOpen(), followed by sceFiosFHPread(), followed by sceFiosFHClose().

sceFiosFileReadSync

Reads bytes from a file without opening it first (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFileReadSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    void *pBuf,
    SceFiosSize length,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to a file to read.
[in] *pBuf*	Buffer to receive file data.
[in] *length*	Requested number of bytes to read.
[in] *offset*	Offset within the file to read from.

Return Values

Non-negative values are the number of bytes read; 0 <= result <= length. Negative values are error codes. Return of zero indicates a read offset starting at or after the end-of-file.

Description

Reads bytes from a file without opening it first (sync). This call is equivalent to sceFiosFHOpen(), followed by sceFiosFHPread(), followed by sceFiosFHClose().

sceFiosFileTruncate

Resizes a file without opening it first.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFileTruncate(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    SceFiosSize length
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to a file to resize.
[in] *length*	New file size.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Resizes a file without opening it first.

sceFiosFileTruncateSync

Resizes a file without opening it first (sync).

Definition

```
#include <fios2_api.h>
int sceFiosFileTruncateSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    SceFiosSize length
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to a file to resize.
[in] *length*	New file size.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Resizes a file without opening it first (sync).

sceFiosFileWrite

Writes bytes to a file without opening it first.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosFileWrite(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    const void *pBuf,
    SceFiosSize length,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to a file to write.
[in] *pBuf*	Buffer with file data to write.
[in] *length*	Requested number of bytes to write.
[in] *offset*	Offset within the file to write to.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Writes bytes to a file without opening it first. This call is equivalent to sceFiosFHOpen(), followed by sceFiosFHPwrite(), followed by sceFiosFHClose(). As with a normal write, this call will overwrite bytes within the file until the end of the file is reached. Once the end of the file is reached any remaining data is appended.

sceFiosFileWriteSync

Writes bytes to a file without opening it first (sync).

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosFileWriteSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    const void *pBuf,
    SceFiosSize length,
    SceFiosOffset offset
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Path to a file to write.
[in] *pBuf*	Buffer with file data to write.
[in] *length*	Requested number of bytes to write.
[in] *offset*	Offset within the file to write to.

Return Values

Non-negative values are the number of bytes written; 0 <= result <= length. Negative values are error codes.

Description

Writes bytes to a file without opening it first (sync). This call is equivalent to sceFiosFHOpen(), followed by sceFiosFHPwrite(), followed by sceFiosFHClose(). As with a normal write, this call will overwrite bytes within the file until the end of the file is reached. Once the end of the file is reached any remaining data is appended.

sceFiosGetAllDHs

Returns an array of all directory handles.

Definition

```
#include <fios2_api.h>
size_t sceFiosGetAllDHs(
    SceFiosDH *pOutArray,
    size_t arraySize
);
```

Arguments

[out] *pOutArray*	Array to receive directory handles. May be `NULL`.
[in] *arraySize*	Number of directory handles that will fit in *pOutArray*. May be 0.

Return Values

The actual number of directory handles that are open.

Description

Returns an array of all directory handles. If the array is larger than required, the unused portion is cleared to 0.

sceFiosGetAllFHs

Returns an array of all file handles.

Definition

```
#include <fios2_api.h>
size_t sceFiosGetAllFHs(
    SceFiosFH *pOutArray,
    size_t arraySize
);
```

Arguments

[out] *pOutArray*	Array to receive file handles. May be `NULL`.
[in] *arraySize*	Number of file handles that will fit in *pOutArray*. May be 0.

Return Values

The actual number of file handles that are open.

Description

Returns an array of all file handles. If the array is larger than required, the unused portion is cleared to 0.

sceFiosGetAllOps

Returns an array of the outstanding operations.

Definition

```
#include <fios2_api.h>
size_t sceFiosGetAllOps(
    SceFiosOp *pOutArray,
    size_t arraySize
);
```

Arguments

[out] *pOutArray*	Array to receive operations. May be `NULL`.
[in] *arraySize*	Number of operations that will fit in *pOutArray*. May be 0.

Return Values

The total number of outstanding operations.

Description

Returns an array of the outstanding operations. If the array is larger than required, the unused portion is cleared to 0.

sceFiosGetDefaultOpAttr

Gets the default operation attributes.

Definition

```
#include <fios2_api.h>
void sceFiosGetDefaultOpAttr(
    SceFiosOpAttr *pOutAttr
);
```

Arguments

[out] *pOutAttr* Returned operation attributes.

Return Values

None

Description

This returns the default operation attributes that will be used for APIs on the current thread. It takes into account both global default attribute settings, and thread-local default attribute settings (when applicable).

sceFiosGetGlobalDefaultOpAttr

Gets the global default operation attributes.

Definition

```
#include <fios2_api.h>
bool sceFiosGetGlobalDefaultOpAttr(
    SceFiosOpAttr *pOutAttr
);
```

Arguments

[out] *pOutAttr* Returned operation attributes.

Return Values

Value	Description
false	Global operation attributes match SCE_FIOS_OPATTR_INITIALIZER.
true	Global operation attributes have been changed to non-default values.

Description

This gets any default operation attributes previously set by sceFiosSetGlobalDefaultOpAttr(). If no default operation attributes have been set, it returns operation attributes that match SCE_FIOS_OPATTR_INITIALIZER.

See Also

sceFiosSetGlobalDefaultOpAttr()

sceFiosGetSuspendCount

Returns FIOS's current suspend count.

Definition

```
#include <fios2_api.h>
uint32_t sceFiosGetSuspendCount();
```

Arguments

None

Return Values

The current suspend count.

Description

Returns FIOS's current suspend count. The suspend count indicates how many times FIOS has been suspended. Upon initialization, FIOS's suspend count is zero. This value is only modified by calls to `sceFiosSuspend()` and `sceFiosResume()`, which should be equally balanced.

sceFiosGetThreadDefaultOpAttr

Gets the thread-local default operation attributes.

Definition

```
#include <fios2_api.h>
bool sceFiosGetThreadDefaultOpAttr(
    SceFiosOpAttr *pOutAttr
);
```

Arguments

[out] *pOutAttr* Returned operation attributes.

Return Values

Value	Description
false	Thread-local operation attributes match SCE_FIOS_OPATTR_INITIALIZER.
true	Thread-local operation attributes have been changed to non-default values.

Description

This gets any default operation attributes previously set by sceFiosSetThreadDefaultOpAttr(). If no default operation attributes have been set, it returns operation attributes that match SCE_FIOS_OPATTR_INITIALIZER.

Notes

Because this function relies on thread-local storage in the compiler, it is not available on all platforms and compilers.

See Also

sceFiosSetThreadDefaultOpAttr(), sceFiosGetGlobalDefaultOpAttr(), sceFiosSetGlobalDefaultOpAttr()

sceFiosInitialize

Initializes FIOS.

Definition

```
#include <fios2_api.h>
int sceFiosInitialize(
    const SceFiosParams *pParameters
);
```

Arguments

[in] *pParameters* A list of parameters for FIOS initialization.

Return Values

Value	Description
SCE_FIOS_OK	Success.
SCE_FIOS_ERROR_BAD_PTR	NULL pointer passed for one of the required storage buffers.
SCE_FIOS_ERROR_BAD_ALIGNMENT	Insufficient alignment passed for one of the required storage buffers.
SCE_FIOS_ERROR_BAD_SIZE	Bad paramsSize, or a bad size (too large or too small) was passed for one of the required storage buffers.

Description

The contents of the param struct are copied to a global inside FIOS, so it doesn't need to be persistent. However, any objects, buffers, or callbacks referenced by the struct must persist for the lifetime of FIOS.

You should always initialize FIOS2. However, system software might use some synchronous FIOS2 API functions before calling sceFiosInitialize(). In this case, the returned error codes might differ from those returned after FIOS2 is initialized. Asynchronous API functions will return SCE_FIOS_OP_INVALID if called before sceFiosInitialize().

Notes

Use SCE_FIOS_PARAMS_INITIALIZER to initialize FIOS *pParameters* with default values. Then, you must provide valid values for the following parameters: *opStorage*, *fhStorage*, *dhStorage*, and *chunkStorage*. The following list describes the specific usage for each of the SceFiosParams initialization parameters:

- *initialized* – Set by FIOS and used only by sceFiosIsInitialized().
- *paramsSize* – Must be set to sizeof(SceFiosParams).
- *pathMax* – If set to 0, FIOS uses SCE_FIOS_PATH_DEFAULT. If greater than SCE_FIOS_PATH_MAX, FIOS uses SCE_FIOS_PATH_MAX.
- *profiling* – Must be set to a valid profiling value; see SceFiosProfilingMask.
- *ioThreadCount* – If less than SCE_FIOS_IO_THREAD_COUNT_MIN, FIOS uses SCE_FIOS_IO_THREAD_COUNT_MIN. If greater than SCE_FIOS_IO_THREAD_COUNT_MAX, FIOS uses SCE_FIOS_IO_THREAD_COUNT_MAX.
- *threadsPerScheduler* – If less than SCE_FIOS_SCHEDULER_THREAD_COUNT_MIN, FIOS uses SCE_FIOS_SCHEDULER_THREAD_COUNT_MIN. If greater than SCE_FIOS_SCHEDULER_THREAD_COUNT_MAX, FIOS uses SCE_FIOS_SCHEDULER_THREAD_COUNT_MAX.

- *extraFlags* – Must be set to zero (0).
- *maxChunk* – If 0, FIOS uses SCE_FIOS_CHUNK_DEFAULT.
- *maxDecompressorThreadCount* – If equal to zero, FIOS uses SCE_FIOS_DECOMPRESSOR_THREAD_COUNT_DEFAULT. If greater than SCE_FIOS_DECOMPRESSOR_THREAD_COUNT_MAX, FIOS uses SCE_FIOS_DECOMPRESSOR_THREAD_COUNT_MAX.
- *reserved1* – Must be set to zero (0).
- *reserved2* – Must be set to zero (0).
- *reserved3* – Must be set to zero (0).
- *reserved4* – Must be set to zero (0).
- *reserved5* – Must be set to zero (0).
- *opStorage* – Must be set to a size of greater than 384+*pathMax* bytes, but less than 512KiB. Must use an alignment that is a multiple of SCE_FIOS_OP_STORAGE_ALIGNMENT.
- *fhStorage* – Must be set to a size of greater than 128+*pathMax* bytes, but less than 512KiB. Must use an alignment that is a multiple of SCE_FIOS_FH_STORAGE_ALIGNMENT.
- *dhStorage* – Must be set to a size of greater than 128+*pathMax* bytes, but less than 512KiB. Must use an alignment that is a multiple of SCE_FIOS_DH_STORAGE_ALIGNMENT.
- *chunkStorage* – Should be at least SCE_FIOS_CHUNK_DEFAULT bytes, but less than SCE_FIOS_CHUNK_STORAGE_SIZE_MAX bytes. Must use an alignment that is a multiple of SCE_FIOS_CHUNK_STORAGE_ALIGNMENT.
- *pVprintf* – If NULL, FIOS uses the platform's default `vprintf` function. Otherwise, must be a valid pointer to a replacement `vprintf` function.
- *pMemcpy* – Must be set to NULL. Note that this initialization value is not currently used.
- *reserved6* – Must be set to zero (0).
- *threadPriority* – Must be set to a valid thread priority value.
- *threadAffinity* – Must be set to a valid thread affinity value.

sceFiosIOFilterAdd

Instantiates a new I/O filter.

Definition

```
#include <fios2_filters.h>
int sceFiosIOFilterAdd(
    int index,
    SceFiosIOFilterCallback pFilterCallback,
    void *pFilterContext
);
```

Arguments

[in] *index*	The index to set in the I/O filter array.
[in] *pFilterCallback*	The callback function used to implement this filter.
[in] *pFilterContext*	A user-defined context pointer which will be passed to the callback.

Return Values

SCE_FIOS_OK for success; SCE_FIOS_ERROR_BAD_INDEX if index is already occupied, if index is less than SCE_FIOS_IOFILTER_INDEX_FIRST, or if index is greater than SCE_FIOS_IOFILTER_INDEX_LAST; or an error code that is specific to the callback.

Description

Instantiates a new I/O filter. The index must not already be occupied. If an error occurs, the I/O filter array will not be modified.

See Also

sceFiosIOFilterRemove()

sceFiosIOFilterCache

An SceFiosIOFilterCallback for a RAM cache.

Definition

```
#include <fios2_filters.h>
void sceFiosIOFilterCache();
```

Arguments

None

Return Values

None

Description

An SceFiosIOFilterCallback for a RAM cache. The *pFilterContext* used with this callback should be an instance of SceFiosRamCacheContext, with at least SCE_FIOS_RAM_CACHE_BUFFER_SIZE bytes (128KiB) of memory available in *pWorkBuffer*.

See Also

sceFiosIOFilterAdd()

sceFiosIOFilterGetInfo

Gets info on an I/O filter.

Definition

```
#include <fios2_filters.h>
int sceFiosIOFilterGetInfo(
    int index,
    SceFiosIOFilterCallback *pOutFilterCallback,
    void **pOutFilterContext
);
```

Arguments

[in] *index*	The index to get in the I/O filter array.
[out] *pOutFilterCallback*	Returns the callback function. May be NULL.
[out] *pOutFilterContext*	Returns the context pointer. May be NULL.

Return Values

`SCE_FIOS_OK` for success, `SCE_FIOS_ERROR_BAD_INDEX` if there is no filter at index, or an error code.

Description

Gets info on an I/O filter.

sceFiosIOFilterPsarcDearchiver

An SceFiosIOFilterCallback for accessing PSARC archives.

Definition

```
#include <fios2_filters.h>
void sceFiosIOFilterPsarcDearchiver();
```

Arguments

None

Return Values

None

Description

An SceFiosIOFilterCallback for accessing PSARC archives. The *pFilterContext* used with this callback should be an instance of SceFiosPsarcDearchiverContext, with at least (3 x block size) bytes of memory available in *pWorkBuffer* where block size is the largest archive block size that the user plans to open.

See Also

sceFiosIOFilterAdd()

sceFiosIOFilterRemove

Removes an I/O filter.

Definition

```
#include <fios2_filters.h>
int sceFiosIOFilterRemove(
    int index
);
```

Arguments

[in] *index* The index in the I/O filter array of the filter to remove.

Return Values

`SCE_FIOS_OK` for success, or an error code.

Description

Removes an I/O filter.

See Also

sceFiosIOFilterAdd()

sceFiosIsIdle

Checks to see whether FIOS is idle.

Definition

```
#include <fios2_api.h>
bool sceFiosIsIdle();
```

Arguments

None

Return Values

Returns `true` if FIOS is idle, `false` otherwise.

Description

Checks to see whether FIOS is idle. FIOS is considered idle if no I/O is being processed. FIOS may be idle even if file handles are open, as long as no I/O is scheduled for any of them. If FIOS is suspended, it is not considered idle if there is pending I/O that has been suspended.

sceFiosIsInitialized

Checks whether or not FIOS has been initialized.

Definition

```
#include <fios2_api.h>
bool sceFiosIsInitialized(
    SceFiosParams *pOutParameters
);
```

Arguments

[out] *pOutParameters*	If FIOS has been initialized, returns a copy of the initialization parameters.

Return Values

Value	Description
false	FIOS has not been initialized.
true	FIOS has been initialized.

Description

Checks whether or not FIOS has been initialized.

sceFiosIsSuspended

Checks to see whether or not FIOS is suspended.

Definition

```
#include <fios2_api.h>
bool sceFiosIsSuspended();
```

Arguments

None

Return Values

Returns `true` if FIOS is suspended, `false` otherwise.

Description

Checks to see whether or not FIOS is suspended.

sceFiosIsValidHandle

Checks to see if a SceFiosFH, SceFiosDH, or SceFiosOp represents a valid FIOS object.

Definition

```
#include <fios2_api.h>
bool sceFiosIsValidHandle(
    SceFiosHandle h
);
```

Arguments

[in] *h* Handle to test.

Return Values

Returns `true` if the handle is valid, `false` otherwise.

Description

Checks to see if a SceFiosFH, SceFiosDH, or SceFiosOp represents a valid FIOS object.

sceFiosOpCancel

Cancels an operation.

Definition

```
#include <fios2_api.h>
void sceFiosOpCancel(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to cancel.

Return Values

None

Description

Cancels an operation.

Notes

This function does NOT wait for the operation to complete. If you need the op to complete, call `sceFiosOpWait()` after cancellation. If the cancel is issued before its associated I/O completes, the op completes with `SCE_FIOS_ERROR_CANCELLED`. Otherwise, a cancelled op can complete with no error (`SCE_FIOS_OK` or the number of bytes read or written) or any of the normal error values.

sceFiosOpDelete

Deletes an asynchronous operation.

Definition

```
#include <fios2_api.h>
void sceFiosOpDelete(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to delete.

Return Values

None

Description

Deletes an asynchronous operation.

sceFiosOpGetActualCount

Returns an operation's actual size in bytes.

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosOpGetActualCount(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to query.

Return Values

Number of bytes actually transferred, or 0.

Description

Returns an operation's actual size in bytes.

sceFiosOpGetAttr

Returns an operation's attributes.

Definition

```
#include <fios2_api.h>
const SceFiosOpAttr *sceFiosOpGetAttr(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to query.

Return Values

Pointer to the operation's attributes. This pointer is only valid until sceFiosOpDelete() is called.

Description

Returns an operation's attributes.

sceFiosOpGetBuffer

Returns an operation's primary buffer.

Definition

```
#include <fios2_api.h>
void *sceFiosOpGetBuffer(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to query.

Return Values

Pointer to the operation's data buffer. For vector I/O such as sceFiosFHReadv(), the first buffer in the vector is returned.

Description

Returns an operation's primary buffer.

sceFiosOpGetError

Returns the error from an operation.

Definition

```
#include <fios2_api.h>
int sceFiosOpGetError(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to query.

Return Values

`SCE_FIOS_ERROR_BAD_OP` if the operation is invalid, `SCE_FIOS_IN_PROGRESS` if the operation isn't done, otherwise the result of the asynchronous operation.

Description

Returns the error from an operation.

Document serial number: 000004892117

sceFiosOpGetOffset

Returns an operation's offset.

Definition

```
#include <fios2_api.h>
SceFiosOffset sceFiosOpGetOffset(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to query.

Return Values

Offset from the original request, or 0.

Description

Returns an operation's offset.

sceFiosOpGetPath

Returns an operation's path.

Definition

```
#include <fios2_api.h>
const char *sceFiosOpGetPath(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to query.

Return Values

Pointer to the operation's path, if any. This pointer is only valid until sceFiosOpDelete() is called.

Description

Returns an operation's path.

sceFiosOpGetRequestCount

Returns an operation's requested size in bytes.

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosOpGetRequestCount(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to query.

Return Values

Number of bytes in the original request, or 0.

Description

Returns an operation's requested size in bytes.

sceFiosOpIsCancelled

Indicates whether an operation has been cancelled.

Definition

```
#include <fios2_api.h>
bool sceFiosOpIsCancelled(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to query.

Return Values

Returns `true` if the operation is valid and has been cancelled, `false` otherwise.

Description

Indicates whether an operation has been cancelled.

sceFiosOpIsDone

Indicates whether an async operation has completed.

Definition

```
#include <fios2_api.h>
bool sceFiosOpIsDone(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to query.

Return Values

Returns `true` if the operation is done or is invalid, `false` if the operation is not done. For an invalid *op*, returning `true` here allows your code to continue and later check for an error value.

Description

Indicates whether an async operation has completed.

sceFiosOpReschedule

Reschedules an operation with a new deadline.

Definition

```
#include <fios2_api.h>
void sceFiosOpReschedule(
    SceFiosOp op,
    SceFiosTime newDeadline
);
```

Arguments

[in] *op*	Operation to reschedule.
[in] *newDeadline*	New deadline.

Return Values

None

Description

Reschedules an operation with a new deadline.

sceFiosOpRescheduleWithPriority

Reschedules an operation with a new deadline and priority.

Definition

```
#include <fios2_api.h>
void sceFiosOpRescheduleWithPriority(
    SceFiosOp op,
    SceFiosTime newDeadline,
    SceFiosPriority newPriority
);
```

Arguments

[in] *op*	Operation to reschedule.
[in] *newDeadline*	New deadline.
[in] *newPriority*	New priority.

Return Values

None

Description

Reschedules an operation with a new deadline and priority.

sceFiosOpSyncWait

Waits for an operation to complete, and then deletes it.

Definition

```
#include <fios2_api.h>
int sceFiosOpSyncWait(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to wait for and delete.

Return Values

Result of the asynchronous operation: `SCE_FIOS_OK` for success, or an error code. For otherwise successful file reads from offsets that are at or after end-of-file, `SCE_FIOS_ERROR_EOF` is returned, and no data is transferred; for otherwise successful file reads that transfer any data, `SCE_FIOS_OK` is returned.

Description

Waits for an operation to complete, and then deletes it.

sceFiosOpSyncWaitForIO

Waits for a read or write operation to complete, and then deletes it.

Definition

```
#include <fios2_api.h>
SceFiosSize sceFiosOpSyncWaitForIO(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to wait for and delete.

Return Values

Result of the asynchronous read or write operation. Non-negative values are the number of bytes read (including zero for read offsets at or after end-of-file); 0 <= result <= (total number of bytes read or written). Negative values are error codes.

Description

Waits for a read or write operation to complete, and then deletes it. This function is used for file sync read and write calls, and is a variant of sceFiosOpSyncWait().

sceFiosOpWait

Waits for an operation to complete.

Definition

```
#include <fios2_api.h>
int sceFiosOpWait(
    SceFiosOp op
);
```

Arguments

[in] *op* Operation to wait for.

Return Values

Result of the asynchronous operation: `SCE_FIOS_OK` for success, or an error code.

Description

Waits for an operation to complete.

sceFiosOpWaitUntil

Waits until a specified time for an operation to complete.

Definition

```
#include <fios2_api.h>
int sceFiosOpWaitUntil(
    SceFiosOp op,
    SceFiosTime deadline
);
```

Arguments

[in] *op*	Operation to wait for.
[in] *deadline*	Deadline at which to give up and stop waiting.

Return Values

`SCE_FIOS_ERROR_TIMEOUT` if the deadline passes before the operation completes, otherwise the result of the asynchronous operation.

Description

Waits until a specified time for an operation to complete.

sceFiosOverlayAdd

Adds an overlay.

Definition

```
#include <fios2_api.h>
int sceFiosOverlayAdd(
    const SceFiosOverlay *pOverlay,
    SceFiosOverlayID *pOutID
);
```

Arguments

[in] *pOverlay*	Description of the overlay to add.
[out] *pOutID*	Filled in with the new SceFiosOverlayID.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Adds an overlay.

Notes

You cannot create a virtual drive with an overlay. The overlay destination cannot be a drive name.

sceFiosOverlayGetInfo

Gets info on an overlay.

Definition

```
#include <fios2_api.h>
int sceFiosOverlayGetInfo(
    SceFiosOverlayID id,
    SceFiosOverlay *pOutOverlay
);
```

Arguments

[in] *id*	The SceFiosOverlayID to query.
[out] *pOutOverlay*	Filled in with the overlay description.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Gets info on an overlay.

sceFiosOverlayGetList

Gets a list of active overlays.

Definition

```
#include <fios2_api.h>
int sceFiosOverlayGetList(
    SceFiosOverlayID *pOutIDs,
    size_t maxIDs,
    size_t *pActualIDs
);
```

Arguments

[out] *pOutIDs*	If not NULL, filled in with a list of SceFiosOverlayIDs.
[in] *maxIDs*	The number of SceFiosOverlayIDs that can fit into the buffer at *pOutIDs*.
[out] *pActualIDs*	If not NULL, filled in with the number of active overlays.

Return Values

SCE_FIOS_OK for success, or an error code. Note that if pOutIDs is NULL and maxIDs is non-zero, this function returns SCE_FIOS_ERROR_BAD_PTR; otherwise, it returns SCE_FIOS_OK.

Description

Gets a list of active overlays.

sceFiosOverlayModify

Modifies an overlay.

Definition

```
#include <fios2_api.h>
int sceFiosOverlayModify(
    SceFiosOverlayID id,
    const SceFiosOverlay *pNewValue
);
```

Arguments

[in] *id*	The SceFiosOverlayID to modify.
[in] *pNewValue*	New overlay description.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Modifies an overlay.

sceFiosOverlayRemove

Removes an overlay.

Definition

```
#include <fios2_api.h>
int sceFiosOverlayRemove(
    SceFiosOverlayID id
);
```

Arguments

[in] *id* The SceFiosOverlayID to remove.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Removes an overlay.

sceFiosOverlayResolveSync

Resolves a path through the active overlays.

Definition

```
#include <fios2_api.h>
int sceFiosOverlayResolveSync(
    int resolveFlag,
    const char *pInPath,
    char *pOutPath,
    size_t maxPath
);
```

Arguments

[in] *resolveFlag*	How to resolve. See SceFiosOverlayResolveMode.
[in] *pInPath*	Path before overlays are applied.
[out] *pOutPath*	Filled in with the path after overlays are applied.
[in] *maxPath*	Size of the buffer at *pOutPath*.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Resolves a path through the active overlays.

Notes

This is a synchronous function, and it might need to issue I/O to check for a file's existence and timestamps. There is currently no asynchronous variant.

sceFiosPathcmp

Compares two path strings using FIOS's active case-folding rules.

Definition

```
#include <fios2_api.h>
int sceFiosPathcmp(
    const char *pA,
    const char *pB
);
```

Arguments

[in] *pA* First path to compare.
[in] *pB* Second path to compare.

Return Values

0 if the paths are equivalent, a negative value if *pA* is lexicographically less than *pB*, positive if *pA* is lexicographically greater than *pB*.

Description

Compares two path strings using FIOS's active case-folding rules.

sceFiosPathncmp

Compares the first *n* characters of two path strings using FIOS's active case-folding rules.

Definition

```
#include <fios2_api.h>
int sceFiosPathncmp(
    const char *pA,
    const char *pB,
    size_t n
);
```

Arguments

[in] *pA*	First path to compare.
[in] *pB*	Second path to compare.
[in] *n*	Compare at most this many characters.

Return Values

`0` if the paths are equivalent, negative if *pA* is lexicographically less than *pB*, positive if *pA* is lexicographically greater than *pB*.

Description

Compares the first *n* characters of two path strings using FIOS's active case-folding rules.

sceFiosPrintf

Prints a message using FIOS's output channel.

Definition

```
#include <fios2_api.h>
int sceFiosPrintf(
    const char *pFormat,
    ...
);
```

Arguments

[in] *pFormat*	Format string.
[in] ...	Additional parameters as in printf.

Return Values

Result from sceFiosVprintf().

Description

Prints a message using FIOS's output channel. This function simply sends its arguments to sceFiosVprintf().

Document serial number: 000004892117

sceFiosRename

Renames a file or directory.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosRename(
    const SceFiosOpAttr *pAttr,
    const char *pOldPath,
    const char *pNewPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pOldPath*	File or directory to move.
[in] *pNewPath*	New name and location of the file or directory.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Renames a file or directory. This function will rename a file or directory within a logical volume, but cannot be used to copy or move files across volumes.

sceFiosRenameSync

Renames a file or directory (sync).

Definition

```
#include <fios2_api.h>
int sceFiosRenameSync(
    const SceFiosOpAttr *pAttr,
    const char *pOldPath,
    const char *pNewPath
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pOldPath*	File or directory to move.
[in] *pNewPath*	New name and location of the file or directory.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Renames a file or directory (sync). This function will rename a file or directory within a logical volume, but cannot be used to copy or move files across volumes. If the *pOldPath* and *pNewPath* values are the same, this function has no effect and returns SCE_FIOS_OK.

sceFiosResolve

Resolves a (path,offset,len) tuple through the filters.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosResolve(
    const SceFiosOpAttr *pAttr,
    const SceFiosTuple *pInTuple,
    SceFiosTuple *pOutTuple
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pInTuple*	(Path,Offset,Length) tuple to resolve.
[out] *pOutTuple*	On successful completion, filled in with the resulting (Path,Offset,Length) tuple. May be the same as *pInTuple*.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Resolves a (path,offset,len) tuple through the filters. This function takes overlays, archives, and other transformations into account.

sceFiosResolveSync

Resolves a (path,offset,len) tuple through the filters (sync).

Definition

```
#include <fios2_api.h>
int sceFiosResolveSync(
    const SceFiosOpAttr *pAttr,
    const SceFiosTuple *pInTuple,
    SceFiosTuple *pOutTuple
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pInTuple*	(Path,Offset,Length) tuple to resolve.
[out] *pOutTuple*	On successful completion, filled in with the resulting (Path,Offset,Length) tuple. May be the same as *pInTuple*.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Resolves a (path,offset,len) tuple through the filters (sync). This function takes overlays, archives, and other transformations into account.

sceFiosResume

Resumes FIOS after suspension.

Definition

```
#include <fios2_api.h>
void sceFiosResume();
```

Arguments

None

Return Values

None

Description

Resumes FIOS after suspension. This call decrements FIOS's suspend count. If the suspend count reaches zero as a result, FIOS is resumed and I/O is started immediately. If the suspend count is already zero, this call has no effect.

sceFiosSetGlobalDefaultOpAttr

Sets the global default operation attributes.

Definition

```
#include <fios2_api.h>
void sceFiosSetGlobalDefaultOpAttr(
    const SceFiosOpAttr *pAttr
);
```

Arguments

[in] *pAttr* New default operation attributes.

Return Values

None

Description

This sets default operation attributes that are used when no other attributes are specified. These global default attributes are only used when an API call is made with a `NULL` attribute pointer, and when there are no thread-local default attributes specified.

The contents of the `SceFiosOpAttr` struct are copied to a global inside FIOS, so it doesn't need to be persistent.

See Also

`sceFiosGetGlobalDefaultOpAttr()`

sceFiosSetThreadDefaultOpAttr

Sets the thread-local default operation attributes.

Definition

```
#include <fios2_api.h>
void sceFiosSetThreadDefaultOpAttr(
    const SceFiosOpAttr *pAttr
);
```

Arguments

[in] *pAttr* New default operation attributes.

Return Values

None

Description

This sets default operation attributes which are used when no other attributes are specified. These thread-local default attributes will only be used when an API call is made with a `NULL` attribute pointer on the current thread.

The contents of the `SceFiosOpAttr` struct are copied to a thread-local variable inside FIOS.

Notes

Because this function relies on thread-local storage in the compiler, it is not available on all platforms and compilers.

See Also

`sceFiosGetThreadDefaultOpAttr()`, `sceFiosGetGlobalDefaultOpAttr()`, `sceFiosSetGlobalDefaultOpAttr()`

sceFiosShutdownAndCancelOps

Shuts down FIOS and cancels any outstanding operations.

Definition

```
#include <fios2_api.h>
void sceFiosShutdownAndCancelOps();
```

Arguments

None

Return Values

None

Description

Shuts down FIOS and cancels any outstanding operations.

sceFiosStat

Returns a full set of status information for a file or directory.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosStat(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    SceFiosStat *pOutStatus
);
```

Arguments

[in] *pAttr*	Operation attributes. May be `NULL`.
[in] *pPath*	Path to query.
[out] *pOutStatus*	Upon successful completion, filled in with a `SceFiosStat` structure.

Return Values

Operation handle. Caller is responsible for deleting this handle with `sceFiosOpDelete()`.

Description

Returns a full set of status information for a file or directory.

Notes

Because it may require more I/O to get all of the information needed by `sceFiosStat()`, you should only use this call if you are looking for information other than size and existence.

sceFiosStatisticsPrint

Prints the current FIOS statistics.

Definition

```
#include <fios2_api.h>
void sceFiosStatisticsPrint();
```

Arguments

None

Return Values

None

Description

Prints the current FIOS statistics.

See Also

sceFiosStatisticsReset().

sceFiosStatisticsReset

Resets FIOS statistics.

Definition

```
#include <fios2_api.h>
void sceFiosStatisticsReset();
```

Arguments

None

Return Values

None

Description

Resets the FIOS statistics to initial values.

See Also

sceFiosStatisticsPrint()

sceFiosStatSync

Returns a full set of status information for a file or directory (sync).

Definition

```
#include <fios2_api.h>
int sceFiosStatSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    SceFiosStat *pOutStatus
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[out] *pPath*	Path to query.
[out] *pOutStatus*	Upon successful completion, filled in with a SceFiosStat structure.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Returns a full set of status information for a file or directory (sync).

Notes

Because it may require more I/O to get all of the information needed by sceFiosStatSync(), you should only use this call if you are looking for information other than size and existence.

sceFiosSuspend

Temporarily suspends FIOS.

Definition

```
#include <fios2_api.h>
void sceFiosSuspend();
```

Arguments

None

Return Values

None

Description

Temporarily suspends FIOS. FIOS may be suspended to temporarily stop all I/O. This call increments FIOS's suspend count; while the suspend count is non-zero, FIOS is suspended. New I/O requests will be accepted while FIOS is suspended, but they will not be processed until the suspend count reaches zero.

sceFiosSync

Flushes changes to a device.

Definition

```
#include <fios2_api.h>
SceFiosOp sceFiosSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    int flag
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Device name.
[in] *flag*	Device-specific flags.

Return Values

Operation handle. Caller is responsible for deleting this handle with sceFiosOpDelete().

Description

Flushes changes to a device. Available only on PlayStation®Vita.

This function might fail if the underlying media doesn't support the sync command. For example, an error may be returned for filesystems on the network.

sceFiosSyncSync

Flushes changes to a device (sync).

Definition

```
#include <fios2_api.h>
int sceFiosSyncSync(
    const SceFiosOpAttr *pAttr,
    const char *pPath,
    int flag
);
```

Arguments

[in] *pAttr*	Operation attributes. May be NULL.
[in] *pPath*	Device name.
[in] *flag*	Device-specific flags. If unknown, use zero.

Return Values

SCE_FIOS_OK for success, or an error code.

Description

Flushes changes to a device (sync). Available only on PlayStation®Vita.

This function might fail if the underlying media doesn't support the sync command. For example, an error may be returned for filesystems on the network.

sceFiosTerminate

Terminates FIOS.

Definition

```
#include <fios2_api.h>
void sceFiosTerminate();
```

Arguments

None

Return Values

None

Description

Terminates FIOS. This call also closes currently open files. This call is not normally necessary. However, it might be helpful if you are planning on fully unloading FIOS from memory.

sceFiosTimeGetCurrent

Gets the current time.

Definition

```
#include <fios2_api.h>
SceFiosTime sceFiosTimeGetCurrent();
```

Arguments

None

Return Values

The current time.

Description

Gets the current time.

sceFiosTimeIntervalFromNanoseconds

Converts nanoseconds to a SceFiosTimeInterval.

Definition

```
#include <fios2_api.h>
SceFiosTimeInterval sceFiosTimeIntervalFromNanoseconds(
    int64_t ns
);
```

Arguments

[in] *ns* Nanoseconds to convert.

Return Values

The equivalent SceFiosTimeInterval.

Description

Converts nanoseconds to a SceFiosTimeInterval.

sceFiosTimeIntervalToNanoseconds

Converts a SceFiosTimeInterval to nanoseconds.

Definition

```
#include <fios2_api.h>
int64_t sceFiosTimeIntervalToNanoseconds(
    SceFiosTimeInterval interval
);
```

Arguments

[in] *interval* Interval to convert.

Return Values

The equivalent number of nanoseconds.

Description

Converts a SceFiosTimeInterval to nanoseconds.

Notes

The result is rounded upward, so an interval representing 1.5 nanoseconds will return the value 2.

sceFiosUpdateParameters

Updates the active FIOS parameters.

Definition

```
#include <fios2_api.h>
void sceFiosUpdateParameters(
    const SceFiosParams *pParameters
);
```

Arguments

[in] *pParameters* New parameters.

Return Values

None

Description

Updates the active FIOS parameters.

Notes

You can only update *profiling*, *maxChunk*, and *pVprintf*. The other startup parameters, such as the buffers that you passed during initialization, cannot be changed.

sceFiosVprintf

Prints a message using FIOS's output channel.

Definition

```
#include <fios2_api.h>
int sceFiosVprintf(
    const char *pFormat,
    va_list ap
);
```

Arguments

[in] *pFormat*	Format string.
[in] *ap*	Additional parameters as in `vprintf`.

Return Values

Result from *pVprintf* or `vprintf`.

Description

Prints a message using FIOS's output channel. This function will use the callback from `SceFiosParams`.*pVprintf*, or the system standard `vprintf` if no callback is supplied.

Document serial number: 000004892117

Callback Functions

SceFiosIOFilterCallback

I/O filter callback.

Definition

```
#include <fios2_filters.h>
typedef void (*SceFiosIOFilterCallback)();
```

Arguments

None

Return Values

None

Description

I/O filter callback.

Notes

The details of the `SceFiosIOFilterCallback` are private and subject to modification until a future release.

SceFiosMemcpyCallback

Callback used for large memcpy operations.

Definition

```
#include <fios2_types.h>
typedef void * (*SceFiosMemcpyCallback)(
    void *dst,
    const void *src,
    size_t len
);
```

Arguments

dst	Destination buffer. Never overlaps the source buffer.
src	Source buffer. Never overlaps the destination buffer.
len	Number of bytes to copy. Always at least 4KiB.

Return Values

The return value is always ignored by FIOS and is provided only for compatibility with the standard prototype for `memcpy`.

Description

Callback used for large memcpy operations.

SceFiosOpCallback

Callback made when an operation completes or is deleted.

Definition

```
#include <fios2_types.h>
typedef int (*SceFiosOpCallback)(
    void *pContext,
    SceFiosOp op,
    SceFiosOpEvent event,
    int err
);
```

Arguments

pContext	User-specified context pointer.
op	Operation that generated this event.
event	Event type.
err	Error from the operation, as returned by sceFiosOpGetError().

Return Values

Returns SCE_FIOS_OK if the event was handled.

Description

Callback made when an operation completes or is deleted. This callback should be quick and not run for any significant length of time (either processing data or waiting for another operation). Doing so may delay other I/O requests.

You may delete the incoming operation as part of handling a completion event, but be aware that this causes your callback to be re-entered with a deletion event.

Notes

For asynchronous calls (such as sceFiosFHOpen), a callback can be called before the asynchronous call returns. A callback can be called from the caller's thread or from a FIOS2 thread.

See Also

SceFiosOpEvents

SceFiosProfileCallback

Callback used for profiling.

Definition

```
#include <fios2_types.h>
typedef void (*SceFiosProfileCallback)(
    const SceFiosProfilingEvent *event
);
```

Arguments

event	The generic event data structure passed to the profile callback.

Return Values

None

Description

Callback used for profiling. This function is called when one of the SceFiosProfilingEventType events completes; when called, the endTime of the event should be very close to the current time.

This callback can be called from the FIOS threads, so there might be a limited amount of stack space.

This callback should be fast and not run for any significant length of time, otherwise I/O could be significantly impacted.

SceFiosVprintfCallback

Callback used for text output.

Definition

```
#include <fios2_types.h>
typedef int (*SceFiosVprintfCallback)(
    const char *fmt,
    va_list ap
);
```

Arguments

fmt	Format as in vprintf.
ap	A *va_list* of additional arguments.

Return Values

The return value is always ignored by FIOS and is provided only for compatibility with the standard prototype for vprintf.

Description

Callback used for text output.

Typedefs

SceFiosBuffer

A buffer in memory.

Definition

```
#include <fios2_types.h>
typedef struct SceFiosBuffer {
    void *pPtr;
    size_t length;
} SceFiosBuffer;
```

Members

pPtr	Pointer to the start of the buffer.
length	Length in bytes.

Description

A buffer in memory.

SceFiosDate

An absolute date and time, like a file timestamp.

Definition

```
#include <fios2_types.h>
typedef uint64_t SceFiosDate;
```

Description

An absolute date and time, like a file timestamp. A `SceFiosDate` contains nanoseconds since the Unix epoch (00:00:00 Jan 1 1970 UTC).

See Also

`SceFiosStat`, `sceFiosDateGetCurrent()`, `sceFiosDateFromComponents()`, `sceFiosDateToComponents()`

SceFiosDH

Directory handle.

Definition

```
#include <fios2_types.h>
typedef SceFiosHandle SceFiosDH;
```

Description

Directory handle. Valid values are always greater than 0. `SCE_FIOS_DH_INVALID` can be used as an initializer.

See Also

`sceFiosDHOpen()`, `sceFiosDHOpenSync()`

SceFiosDirEntry

Directory entry.

Definition

```
#include <fios2_types.h>
typedef struct SceFiosDirEntry {
    SceFiosOffset fileSize;
    uint32_t statFlags;
    uint16_t nameLength;
    uint16_t fullPathLength;
    uint16_t offsetToName;
    uint16_t reserved[3];
    char fullPath[SCE_FIOS_PATH_MAX];
} SceFiosDirEntry;
```

Members

fileSize	File size in bytes.
statFlags	File status flags, from SceFiosStatusFlags.
nameLength	Name length, for convenience.
fullPathLength	Full path length, for convenience.
offsetToName	Offset from start of full path to start of filename.
reserved	Reserved, set to 0.
fullPath	Full path.

Description

Directory entry.

See Also

sceFiosDHRead(), sceFiosDHReadSync()

SceFiosFH

File handle.

Definition

```
#include <fios2_types.h>
typedef SceFiosHandle SceFiosFH;
```

Description

File handle. Valid values are always greater than 0. `SCE_FIOS_FH_INVALID` can be used as an initializer.

See Also

`sceFiosFHOpen()`, `sceFiosFHOpenSync()`

SceFiosHandle

Generic handle type.

Definition

```
#include <fios2_types.h>
typedef int32_t SceFiosHandle;
```

Description

Generic handle type. Valid values are always greater than 0. `SCE_FIOS_HANDLE_INVALID` can be used as an initializer.

See Also

SceFiosOp, SceFiosFH, SceFiosDH

SceFiosIoFilterIndex

I/O filter index values.

Definition

```
#include <fios2_filters.h>
typedef enum SceFiosIoFilterIndex {
    SCE_FIOS_IOFILTER_INDEX_FIRST = 0,
    SCE_FIOS_IOFILTER_INDEX_LAST = 255
} SceFiosIoFilterIndex;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_IOFILTER_INDEX_FIRST	0	Filters with this index will be applied first.
SCE_FIOS_IOFILTER_INDEX_LAST	255	Filters with this index will be applied last.

Description

I/O filter index values.

See Also

sceFiosIOFilterAdd(), sceFiosIOFilterGetInfo(), sceFiosIOFilterRemove()

SceFiosIoProfileData

Profiling data for I/O events.

Definition

```
#include <fios2_types.h>
typedef struct SceFiosIoProfileData {
    SceFiosOffset offset;
    SceFiosOffset LBA;
    SceFiosSize length;
    SceFiosTime deadline;
    const char *path;
    uint16_t chunkID;
    int8_t priority;
    int8_t reserved[1];
} SceFiosIoProfileData;
```

Members

offset	Offset in the file for the request (for read or write events).
LBA	LBA of the request. The most significant bit is set if this is a generated LBA; if set to SCE_FIOS_INVALID_LBA no LBA is available.
length	Length of the request (for read or write events).
deadline	Deadline of the operation, see SceFiosOpAttr.
path	Path to the file for this event.
chunkID	Id of the chunk.
priority	Priority of the operation, see SceFiosOpAttr.
reserved	Reserved (for padding).

Description

Profiling data for I/O events.

SceFiosIOThreadCount

The number of I/O schedulers.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosIOThreadCount {
    SCE_FIOS_IO_THREAD_COUNT_MIN = 1,
    SCE_FIOS_IO_THREAD_COUNT_MAX = 2
} SceFiosIOThreadCount;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_IO_THREAD_COUNT_MIN	1	The minimum number of I/O threads allowed.
SCE_FIOS_IO_THREAD_COUNT_MAX	2	The maximum number of I/O threads allowed. Applies to PlayStation®Vita only.

Description

The number of I/O schedulers.

Notes

For all platforms, if the thread count is 1, all I/O executes on one thread. For each *ioThreadCount*, there is an associated scheduler queue, and for each queue, there are *threadsPerScheduler* threads. For PlayStation®Vita, if the thread count is 2, thread 1 is used for host I/O, and thread 0 is used for everything else. For Windows, one thread is used for all I/O.

See Also

SceFiosParams, sceFiosInitialize()

SceFiosOffset

A 64-bit signed integer used for offsets within a file.

Definition

```
#include <fios2_types.h>
typedef int64_t SceFiosOffset;
```

Description

A 64-bit signed integer used for offsets within a file.

SceFiosOp

Operation handle.

Definition

```
#include <fios2_types.h>
typedef SceFiosHandle SceFiosOp;
```

Description

Operation handle. Valid values are always greater than 0. `SCE_FIOS_OP_INVALID` can be used as an initializer.

See Also

`sceFiosOpWait()`, `sceFiosOpIsDone()`, `sceFiosOpGetError()`

SceFiosOpAttr

Operation attributes.

Definition

```
#include <fios2_types.h>
typedef struct SceFiosOpAttr {
    SceFiosTime deadline;
    SceFiosOpCallback pCallback;
    void *pCallbackContext;
    int32_t priority:8;
    uint32_t opflags:24;
    uint32_t userTag;
    void *userPtr;
    void *pReserved;
} SceFiosOpAttr;
```

Members

deadline	Deadline. 0 means *use default deadline* of current time plus 300 seconds, SCE_FIOS_TIME_EARLIEST means *as soon as possible*, and SCE_FIOS_TIME_LATEST means *as late as possible*.
pCallback	Callback for the operation (may be NULL).
pCallbackContext	Callback context pointer (may be NULL).
priority	Priority. 0 is normal priority, negative values are low priority, and positive values are high priority.
opflags	Flags to control execution; see SceFiosOpFlags.
userTag	Tag for your use.
userPtr	Pointer for your use.
pReserved	Reserved, set to 0.

Description

Operation attributes.

SceFiosOpenFlags

Open flags.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosOpenFlags {
    SCE_FIOS_O_READ = (1U<<0),
    SCE_FIOS_O_RDONLY = SCE_FIOS_O_READ,
    SCE_FIOS_O_WRITE = (1U<<1),
    SCE_FIOS_O_WRONLY = SCE_FIOS_O_WRITE,
    SCE_FIOS_O_RDWR = (SCE_FIOS_O_READ | SCE_FIOS_O_WRITE),
    SCE_FIOS_O_APPEND = (1U<<2),
    SCE_FIOS_O_CREAT = (1U<<3),
    SCE_FIOS_O_TRUNC = (1U<<4),
    SCE_FIOS_O_EDATA = (1U<<8),
    SCE_FIOS_O_SDATA = (1U<<9),
    SCE_FIOS_O_SCEEDRM = (1U<<10),
    SCE_FIOS_O_DIRECT = (1U<<12)
} SceFiosOpenFlags;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_O_READ	(1U<<0)	Open with read privileges.
SCE_FIOS_O_RDONLY	SCE_FIOS_O_READ	Synonym for SCE_FIOS_O_READ, used to signify read-only.
SCE_FIOS_O_WRITE	(1U<<1)	Open with write privileges.
SCE_FIOS_O_WRONLY	SCE_FIOS_O_WRITE	Synonym for SCE_FIOS_O_WRITE, used to signify write-only.
SCE_FIOS_O_RDWR	(SCE_FIOS_O_READ | SCE_FIOS_O_WRITE)	Synonym for SCE_FIOS_O_READ and SCE_FIOS_O_WRITE together, used to signify read-write.
SCE_FIOS_O_APPEND	(1U<<2)	Write-append-only.
SCE_FIOS_O_CREAT	(1U<<3)	Create file if it doesn't exist.
SCE_FIOS_O_TRUNC	(1U<<4)	Truncate file if it already exists.
SCE_FIOS_O_EDATA	(1U<<8)	Open an EDATA file.
SCE_FIOS_O_SDATA	(1U<<9)	Open an SDATA file.
SCE_FIOS_O_SCEEDRM	(1U<<10)	Open a SCEE DRM file.
SCE_FIOS_O_DIRECT	(1U<<12)	Minimize use of the system cache during IO. This flag has no affect on the PlayStation®Vita.

Description

Open flags.

Notes

If neither SCE_FIOS_O_READ nor SCE_FIOS_O_WRITE is specified, FIOS2 assumes SCE_FIOS_O_READ and opens the file in read-only mode.

See Also

SceFiosOpenParams, sceFiosFHOpen()

SceFiosOpenParams

File open parameters.

Definition

```
#include <fios2_types.h>
typedef struct SceFiosOpenParams {
    uint32_t openFlags:16;
    uint32_t opFlags:16;
    uint32_t reserved;
    SceFiosBuffer buffer;
} SceFiosOpenParams;
```

Members

openFlags	Open flags, from SceFiosOpenFlags.
opFlags	Op flags cache, for internal use only.
reserved	Reserved, set to 0.
buffer	Optional I/O buffer.

Description

File open parameters.

See Also

sceFiosFHOpen(), sceFiosFHOpenSync()

SceFiosOpEvent

Events used by SceFiosOpCallback.

Definition

```
#include <fios2_types.h>
typedef uint8_t SceFiosOpEvent;
```

Description

Events used by SceFiosOpCallback.

SceFiosOpEvents

Op callback events.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosOpEvents {
    SCE_FIOS_OPEVENT_COMPLETE = 1,
    SCE_FIOS_OPEVENT_DELETE = 2
} SceFiosOpEvents;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_OPEVENT_COMPLETE	1	This operation has completed, either successfully or with an error.
SCE_FIOS_OPEVENT_DELETE	2	This operation is being deleted.

Description

Op callback events.

See Also

SceFiosOpCallback

SceFiosOpFlags

Op flags.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosOpFlags {
    SCE_FIOS_OPFLAG_IMMED = (1<<0),
    SCE_FIOS_OPFLAG_DONTUSECACHE = (1<<1),
    SCE_FIOS_OPFLAG_DONTFILLRAMCACHE = (1<<2),
    SCE_FIOS_OPFLAG_DONTFILLDISKCACHE = (1<<3),
    SCE_FIOS_OPFLAG_CACHEPERSIST = (1<<4),
    SCE_FIOS_OPFLAG_UNCACHEDBUFFER = (1<<5),
    SCE_FIOS_OPFLAG_NONDMABUFFER = (1<<6),
    SCE_FIOS_OPFLAG_PERSISTENTPATH = (1<<7),
    SCE_FIOS_OPFLAG_DONTFILLCACHE = (SCE_FIOS_OPFLAG_DONTFILLRAMCACHE |
    SCE_FIOS_OPFLAG_DONTFILLDISKCACHE),
    SCE_FIOS_OPFLAG_NOCACHE = (SCE_FIOS_OPFLAG_DONTFILLCACHE |
    SCE_FIOS_OPFLAG_DONTUSECACHE)
} SceFiosOpFlags;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_OPFLAG_IMMED	(1<<0)	Set to allow synchronous open, close, or stat related API calls to be immediately executed on the caller's thread, rather than being enqueued for possible execution later. This flag is useful when there are many ops in-flight, and an open or stat call must happen as soon as possible. It is also useful when you know that the ops should be nearly immediate. Note that for a close call, if any ops are using the file handle passed to the close, the close is enqueued for execution after dependent ops have completed (and thus does not run immediately).
SCE_FIOS_OPFLAG_DONTUSECACHE	(1<<1)	Ignore cache-hits when executing this operation.
SCE_FIOS_OPFLAG_DONTFILLRAMCACHE	(1<<2)	Don't cache-fill to any RAM-based caches.
SCE_FIOS_OPFLAG_DONTFILLDISKCACHE	(1<<3)	Don't cache-fill to any disk-based caches. This flag only applies to systems which have a disk cache.

Macro	Value	Description
SCE_FIOS_OPFLAG_CACHEPERSIST	(1<<4)	Flag the read data as non-evictable in any disk-based caches.
SCE_FIOS_OPFLAG_UNCACHEDBUFFER	(1<<5)	Buffers for this I/O are in uncached memory such as VRAM.
SCE_FIOS_OPFLAG_NONDMABUFFER	(1<<6)	Buffers for this I/O are in non-DMAable memory.
SCE_FIOS_OPFLAG_PERSISTENTPATH	(1<<7)	The caller guarantees that the path string will remain valid until the operation is complete.
SCE_FIOS_OPFLAG_DONTFILLCACHE	(SCE_FIOS_OPFLAG_DONTFILLRAMCACHE | SCE_FIOS_OPFLAG_DONTFILLDISKCACHE)	Don't cache-fill to any caches.
SCE_FIOS_OPFLAG_NOCACHE	(SCE_FIOS_OPFLAG_DONTFILLCACHE | SCE_FIOS_OPFLAG_DONTUSECACHE)	Ignore caches entirely when executing this I/O: no cache-hits and no cache-fills.

Description

Op flags.

See Also

SceFiosOpAttr

SceFiosOverlay

Overlay description.

Definition

```
#include <fios2_types.h>
typedef struct SceFiosOverlay {
    uint8_t type;
    uint8_t order;
    uint8_t reserved[10];
    SceFiosOverlayID id;
    char dst[SCE_FIOS_OVERLAY_POINT_MAX];
    char src[SCE_FIOS_OVERLAY_POINT_MAX];
} SceFiosOverlay;
```

Members

type	Overlay type, from SceFiosOverlayType.
order	Search order. Lower orders are applied first, and overlays with the same order are applied from newest to oldest.
reserved	Reserved. Set to 0.
id	Overlay ID (read-only).
dst	Destination path. This is the path that is modified or created. The destination cannot be a drive name; you cannot create a virtual drive with an overlay.
src	Source path. This is the path that is merged with (or replaces) *dst*.

Description

Overlay description.

See Also

sceFiosOverlayAdd(), sceFiosOverlayGetInfo(), sceFiosOverlayModify()

SceFiosOverlayID

A handle to an overlay layer.

Definition

```
#include <fios2_types.h>
typedef int32_t SceFiosOverlayID;
```

Description

A handle to an overlay layer.

SceFiosOverlayLimits

Overlay limits.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosOverlayLimits {
    SCE_FIOS_OVERLAY_MAX_OVERLAYS = 64,
    SCE_FIOS_OVERLAY_POINT_MAX = 292
} SceFiosOverlayLimits;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_OVERLAY_MAX_OVERLAYS	64	Maximum number of overlays supported.
SCE_FIOS_OVERLAY_POINT_MAX	292	Maximum path length for overlay *src* and *dst*.

Description

Overlay limits.

SceFiosOverlayOrder

Overlay search ordering.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosOverlayOrder {
    SCE_FIOS_OVERLAY_ORDER_USER_FIRST = 0,
    SCE_FIOS_OVERLAY_ORDER_USER_LAST = 127
} SceFiosOverlayOrder;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_OVERLAY_ORDER_USER_FIRST	0	Overlays with this order will be applied first.
SCE_FIOS_OVERLAY_ORDER_USER_LAST	127	Overlays with this order will be applied last.

Description

Overlay search ordering.

See Also

SceFiosOverlay

SceFiosOverlayResolveMode

Overlay resolve mode.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosOverlayResolveMode {
    SCE_FIOS_OVERLAY_RESOLVE_FOR_READ = 0,
    SCE_FIOS_OVERLAY_RESOLVE_FOR_WRITE = 1
} SceFiosOverlayResolveMode;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_OVERLAY_RESOLVE_FOR_READ	0	Resolves as if reading an existing file or directory.
SCE_FIOS_OVERLAY_RESOLVE_FOR_WRITE	1	Resolves as if writing a new file or directory.

Description

Overlay resolve mode.

See Also

sceFiosOverlayResolveSync()

SceFiosOverlayType

Overlay types.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosOverlayType {
    SCE_FIOS_OVERLAY_TYPE_OPAQUE = 0,
    SCE_FIOS_OVERLAY_TYPE_TRANSLUCENT = 1,
    SCE_FIOS_OVERLAY_TYPE_NEWER = 2,
    SCE_FIOS_OVERLAY_TYPE_WRITABLE = 3
} SceFiosOverlayType;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_OVERLAY_TYPE_OPAQUE	0	*src* replaces *dst*. All accesses to *dst* are redirected to *src*.
SCE_FIOS_OVERLAY_TYPE_TRANSLUCENT	1	*src* merges with *dst*. Reads check *src* first, then *dst*. Writes go to *dst*.
SCE_FIOS_OVERLAY_TYPE_NEWER	2	*src* merges with *dst*. Reads check both *src* and *dst*, and use whichever has the most recent modification time. If both *src* and *dst* have the same modification time, *dst* is used. If no file exists at *src* or *dst*, *dst* is used; if no file exists at *dst*, but a file exists at *src*, *src* is used. Writes go to *dst*.
SCE_FIOS_OVERLAY_TYPE_WRITABLE	3	*src* merges with *dst*. Reads check *src* first, then *dst*. Writes go to *src*.

Description

Overlay types.

See Also

SceFiosOverlay

SceFiosParams

Initialization parameters.

Definition

```
#include <fios2_types.h>
typedef struct SceFiosParams {
    uint32_t initialized:1;
    uint32_t paramsSize:15;
    uint32_t pathMax:16;
    uint32_t profiling;
    uint32_t ioThreadCount;
    uint32_t threadsPerScheduler;
    uint32_t extraFlag1:1;
    uint32_t extraFlags:31;
    uint32_t maxChunk;
    uint8_t maxDecompressorThreadCount;
    uint8_t reserved1;
    uint8_t reserved2;
    uint8_t reserved3;
    intptr_t reserved4;
    intptr_t reserved5;
    SceFiosBuffer opStorage;
    SceFiosBuffer fhStorage;
    SceFiosBuffer dhStorage;
    SceFiosBuffer chunkStorage;
    SceFiosVprintfCallback pVprintf;
    SceFiosMemcpyCallback pMemcpy;
    SceFiosProfileCallback pProfileCallback;
    int threadPriority[SCE_FIOS_THREAD_TYPES];
    int threadAffinity[SCE_FIOS_THREAD_TYPES];
} SceFiosParams;
```

Members

initialized	Will be set to 1 after FIOS has been initialized.
paramsSize	Set to `sizeof(SceFiosParams)`.
pathMax	Maximum path length. Smaller values reduce memory consumption.
profiling	Flags to enable profiling. See SceFiosProfilingMask.
ioThreadCount	Number of threads used for I/O. See SceFiosIOThreadCount.
threadsPerScheduler	Number of threads used for each non-callback scheduler. See SceFiosSchedulerThreadCount.
extraFlag1	Reserved additional flag. Set to 0.
extraFlags	Additional flags. Set to 0.
maxChunk	Maximum chunk size. Reads or writes larger than this will be broken up into multiple chunks, and other I/O is eligible to be scheduled in between. If set to 0, SCE_FIOS_CHUNK_DEFAULT will be used.
maxDecompressorThreadCount	The number of decompression threads to create. If set to 0, SCE_FIOS_DECOMPRESSOR_THREAD_COUNT_DEFAULT will be used. If larger than SCE_FIOS_DECOMPRESSOR_THREAD_COUNT_MAX, SCE_FIOS_DECOMPRESSOR_THREAD_COUNT_MAX will be used.
reserved1	Reserved. Set to 0.
reserved2	Reserved. Set to 0.
reserved3	Reserved. Set to 0.

reserved4	Reserved. Set to 0.
reserved5	Reserved. Set to 0.
opStorage	Memory to use for ops. Minimum size is SCE_FIOS_OP_STORAGE_SIZE(1, pathMax), maximum size is SCE_FIOS_OP_STORAGE_SIZE(SCE_FIOS_MAX_HANDLE_ELEMENTS, pathMax).
fhStorage	Memory to use for file handles. Minimum size is SCE_FIOS_FH_STORAGE_SIZE(1, pathMax), maximum size is SCE_FIOS_FH_STORAGE_SIZE(SCE_FIOS_MAX_HANDLE_ELEMENTS, pathMax).
dhStorage	Memory to use for directory handles. Minimum size is SCE_FIOS_DH_STORAGE_SIZE(1, pathMax), maximum size is SCE_FIOS_DH_STORAGE_SIZE(SCE_FIOS_MAX_HANDLE_ELEMENTS, pathMax).
chunkStorage	Memory to use for chunks. Maximum size is SCE_FIOS_CHUNK_STORAGE_SIZE_MAX.
pVprintf	All TTY output from FIOS is funneled through this. Default is standard `vprintf`.
pMemcpy	Used for large (>4KiB) memory copies. Default is standard `memcpy`. This value is not currently used.
pProfileCallback	Call this function when profile callback events complete; see SceFiosProfileCallback.
threadPriority	Thread priorities. See SceFiosThreadType.
threadAffinity	Thread affinity. See SceFiosThreadType.

Description

Initialization parameters. See sceFiosInitialize for information about how to set each of the `sceFiosParams` initialization parameters. See sceFiosIsInitialized for information about how to determine if FIOS is initialized. See sceFiosUpdateParameters for information about how to update the parameters.

Notes

For each *ioThreadCount*, there is an associated scheduler queue, and for each queue, there are *threadsPerScheduler* threads.

See Also

sceFiosInitialize(), sceFiosIsInitialized(), sceFiosUpdateParameters()

SceFiosPriority

I/O priority.

Definition

```
#include <fios2_types.h>
typedef int8_t SceFiosPriority;
```

Description

I/O priority.

See Also

SceFiosOpAttr, SCE_FIOS_PRIO_MIN, SCE_FIOS_PRIO_DEFAULT, SCE_FIOS_PRIO_MAX

SceFiosProfilingEvent

A profiling event.

Definition

```
#include <fios2_types.h>
typedef struct SceFiosProfilingEvent {
    SceFiosTime startTime;
    SceFiosTime endTime;
    const void *data;
    int8_t eventType;
    int8_t reserved[3];
} SceFiosProfilingEvent;
```

Members

startTime	Start time of event. For Windows, the TSC for the start of the event; for PlayStation®Vita, a standard SceFiosTime value.
endTime	End time of event. For Windows, the TSC for the end of the event; for PlayStation®Vita, a standard SceFiosTime value.
data	For scheduler events a pointer to SceFiosSchedulerProfileData; for I/O events a pointer to SceFiosIoProfileData.
eventType	See SceFiosProfilingEventType.
reserved	Reserved (for padding).

Description

A profiling event.

SceFiosProfilingEventType

Profiling event type.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosProfilingEventType {
    SCE_FIOS_PROFILE_EVENT_OPEN = 0,
    SCE_FIOS_PROFILE_EVENT_READ = 1,
    SCE_FIOS_PROFILE_EVENT_WRITE = 2,
    SCE_FIOS_PROFILE_EVENT_CLOSE = 3,
    SCE_FIOS_PROFILE_EVENT_STAT = 4,
    SCE_FIOS_PROFILE_EVENT_SYNC = 5,
    SCE_FIOS_PROFILE_EVENT_OPENDIR = 6,
    SCE_FIOS_PROFILE_EVENT_READDIR = 7,
    SCE_FIOS_PROFILE_EVENT_CLOSEDIR = 8,
    SCE_FIOS_PROFILE_EVENT_SCHEDULETREE = 9,
    SCE_FIOS_PROFILE_EVENT_UNSCHEDULETREE = 10,
    SCE_FIOS_PROFILE_EVENT_ADDTOSCHEDULE = 11,
    SCE_FIOS_PROFILE_EVENT_REMOVEFROMSCHEDULE = 12,
    SCE_FIOS_PROFILE_EVENT_EXECUTE = 13
} SceFiosProfilingEventType;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_PROFILE_EVENT_OPEN	0	I/O: Open file event.
SCE_FIOS_PROFILE_EVENT_READ	1	I/O: Read from file event.
SCE_FIOS_PROFILE_EVENT_WRITE	2	I/O: Write to file event.
SCE_FIOS_PROFILE_EVENT_CLOSE	3	I/O: Close file event.
SCE_FIOS_PROFILE_EVENT_STAT	4	I/O: File stat event.
SCE_FIOS_PROFILE_EVENT_SYNC	5	I/O: File sync (or flush) event.
SCE_FIOS_PROFILE_EVENT_OPENDIR	6	I/O: Open directory event.
SCE_FIOS_PROFILE_EVENT_READDIR	7	I/O: Read directory event.
SCE_FIOS_PROFILE_EVENT_CLOSEDIR	8	I/O: Close directory event.
SCE_FIOS_PROFILE_EVENT_SCHEDULETREE	9	Scheduler: Schedule tree event.
SCE_FIOS_PROFILE_EVENT_UNSCHEDULETREE	10	Scheduler: Unschedule tree event.
SCE_FIOS_PROFILE_EVENT_ADDTOSCHEDULE	11	Scheduler: Add a chunk to scheduler.
SCE_FIOS_PROFILE_EVENT_REMOVEFROMSCHEDULE	12	Scheduler: Remove a chunk from scheduler.
SCE_FIOS_PROFILE_EVENT_EXECUTE	13	Scheduler: Execute a chunk.

Description

Profiling event type.

SceFiosProfilingMask

Profiling mask.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosProfilingMask {
    SCE_FIOS_PROFILE_API = (1U<<0),
    SCE_FIOS_PROFILE_SCHEDULER = (1U<<1),
    SCE_FIOS_PROFILE_OP = (1U<<2),
    SCE_FIOS_PROFILE_CHUNK = (1U<<3),
    SCE_FIOS_PROFILE_IO = (1U<<4),
    SCE_FIOS_PROFILE_CACHE = (1U<<5),
    SCE_FIOS_PROFILE_DEARCHIVER = (1U<<6),
    SCE_FIOS_PROFILE_OVERLAY = (1U<<7),
    SCE_FIOS_PROFILE_ALL = 0xFFFFFFFFU
} SceFiosProfilingMask;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_PROFILE_API	(1U<<0)	Enable API profiling.
SCE_FIOS_PROFILE_SCHEDULER	(1U<<1)	Enable scheduler profiling.
SCE_FIOS_PROFILE_OP	(1U<<2)	Enable op profiling.
SCE_FIOS_PROFILE_CHUNK	(1U<<3)	Enable chunk profiling.
SCE_FIOS_PROFILE_IO	(1U<<4)	Enable I/O profiling.
SCE_FIOS_PROFILE_CACHE	(1U<<5)	Enable cache profiling.
SCE_FIOS_PROFILE_DEARCHIVER	(1U<<6)	Enable dearchiver profiling.
SCE_FIOS_PROFILE_OVERLAY	(1U<<7)	Enable overlay profiling.
SCE_FIOS_PROFILE_ALL	0xFFFFFFFFU	Enable all profiling.

Description

Profiling mask.

See Also

SceFiosParams, sceFiosInitialize()

SceFiosPsarcDearchiverContext

Context for the sceFiosIOFilterPsarcDearchiver.

Definition

```
#include <fios2_filters.h>
typedef struct SceFiosPsarcDearchiverContext {
    size_t sizeOfContext;
    size_t workBufferSize;
    void *pWorkBuffer;
    intptr_t flags;
    intptr_t reserved[3];
} SceFiosPsarcDearchiverContext;
```

Members

sizeOfContext	Set to `sizeof(SceFiosPsarcDearchiverContext);` used for compatibility across versions.
workBufferSize	Size of the memory pointed to by *pWorkBuffer*.
pWorkBuffer	Pointer to a buffer containing enough bytes to store (3 x block size) of the largest archive block size that the user opens.
flags	Behavior flags, from SceFiosPsarcDearchiverFlags.
reserved	Reserved for the implementation. Set to 0.

Description

Context for the sceFiosIOFilterPsarcDearchiver().

SceFiosPsarcDearchiverFlags

Dearchiver flags for SceFiosPsarcDearchiverContext.*flags*.

Definition

```
#include <fios2_filters.h>
typedef enum SceFiosPsarcDearchiverFlags {
    SCE_FIOS_PSARC_DEARCHIVER_MOUNT_TRANSLUCENT = (1<<0)
} SceFiosPsarcDearchiverFlags;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_PSARC_DEARCHIVER_MOUNT_TRANSLUCENT	(1<<0)	Set this flag to make mount points translucent, so that previously existing files underneath a mount point will be accessible. If this flag is not set, mounts will be opaque, and previously existing files at the mount point will become inaccessible.

Description

Dearchiver flags for SceFiosPsarcDearchiverContext.*flags*.

See Also

SceFiosPsarcDearchiverContext

SceFiosRamCacheContext

Context for the sceFiosIOFilterCache.

Definition

```
#include <fios2_filters.h>
typedef struct SceFiosRamCacheContext {
    size_t sizeOfContext;
    size_t workBufferSize;
    size_t blockSize;
    void *pWorkBuffer;
    const char *pPath;
    intptr_t flags;
    intptr_t reserved[3];
} SceFiosRamCacheContext;
```

Members

sizeOfContext	Set to `sizeof(SceFiosRamCacheContext);` used for compatibility across versions.
workBufferSize	Size of the memory pointed to by *pWorkBuffer*.
blockSize	Size of individual cache block. Best if multiple of chunk size.
pWorkBuffer	Pointer to a buffer of at least SCE_FIOS_RAM_CACHE_BUFFER_SIZE bytes.
pPath	Path associated with this cache filter. May be NULL.
flags	Behavior flags, set to 0 for now.
reserved	Reserved for the implementation. Set to 0.

Description

Context for the sceFiosIOFilterCache.

SceFiosSchedulerProfileData

Profiling data for Scheduler events.

Definition

```
#include <fios2_types.h>
typedef struct SceFiosSchedulerProfileData {
    intptr_t threadId;
    uint16_t chunkID;
    uint8_t chunkType;
    int8_t reserved[1];
} SceFiosSchedulerProfileData;
```

Members

threadId	threadId for the event. This can be a user's thread, FIOS2 callback scheduler thread, or FIOS2 scheduler thread.
chunkID	Id of the chunk.
chunkType	Type of chunk.
reserved	Reserved (for padding).

Description

Profiling data for Scheduler events.

SceFiosSchedulerThreadCount

Presets for the number of non-callback scheduler threads.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosSchedulerThreadCount {
    SCE_FIOS_SCHEDULER_THREAD_COUNT_MIN = 1,
    SCE_FIOS_SCHEDULER_THREAD_COUNT_MAX = 2,
    SCE_FIOS_SCHEDULER_THREAD_COUNT_DEFAULT = 1
} SceFiosSchedulerThreadCount;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_SCHEDULER_THREAD_COUNT_MIN	1	The minimum number of non-callback scheduler threads allowed.
SCE_FIOS_SCHEDULER_THREAD_COUNT_MAX	2	The maximum number of non-callback scheduler threads allowed (for PlayStation®Vita).
SCE_FIOS_SCHEDULER_THREAD_COUNT_MAX	3	The maximum number of non-callback scheduler threads allowed (for Windows).
SCE_FIOS_SCHEDULER_THREAD_COUNT_DEFAULT	1	The default number of non-callback scheduler threads.

Description

Presets for the number of non-callback scheduler threads.

See Also

SceFiosParams, sceFiosInitialize()

SceFiosSize

A 64-bit signed integer used for file sizes.

Definition

```
#include <fios2_types.h>
typedef int64_t SceFiosSize;
```

Description

A 64-bit signed integer used for file sizes.

SceFiosStat

File status.

Definition

```
#include <fios2_types.h>
typedef struct SceFiosStat {
    SceFiosOffset fileSize;
    SceFiosDate accessDate;
    SceFiosDate modificationDate;
    SceFiosDate creationDate;
    uint32_t statFlags;
    uint32_t reserved;
    int64_t uid;
    int64_t gid;
    int64_t dev;
    int64_t ino;
    int64_t mode;
} SceFiosStat;
```

Members

fileSize	File size in bytes.
accessDate	Last accessed date.
modificationDate	Modification date.
creationDate	Creation date.
statFlags	File status flags, from SceFiosStatusFlags.
reserved	Reserved, set to 0.
uid	User ID. Not supported on PlayStation®Vita.
gid	Group ID. Not supported on PlayStation®Vita.
dev	Device number. Not supported on PlayStation®Vita.
ino	Inode number. Not supported on PlayStation®Vita.
mode	File mode.

Description

File status. These values are obtained from the underlying filesystem. The *fileSize*, *statFlags*, and *modificationDate* fields are always valid. Other fields may not be supported due to limitations either in the OS or the filesystem. When a field is not supported, it will be set to 0.

See Also

sceFiosStat(), sceFiosStatSync(), sceFiosFHStat(), sceFiosFHStatSync()

SceFiosStatusFlags

File status flags.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosStatusFlags {
    SCE_FIOS_STATUS_DIRECTORY = (1<<0),
    SCE_FIOS_STATUS_READABLE = (1<<1),
    SCE_FIOS_STATUS_WRITABLE = (1<<2)
} SceFiosStatusFlags;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_STATUS_DIRECTORY	(1<<0)	Item is a directory.
SCE_FIOS_STATUS_READABLE	(1<<1)	Item is readable.
SCE_FIOS_STATUS_WRITABLE	(1<<2)	Item is writable.

Description

File status flags.

See Also

SceFiosStat, SceFiosDirEntry, sceFiosStat(), sceFiosStatSync(), sceFiosFHStat(), sceFiosFHStatSync(), sceFiosDHRead(), sceFiosDHReadSync(), sceFiosChangeStat(), sceFiosChangeStatSync()

SceFiosThreadType

Thread types.

Definition

```
#include <fios2_platform_common.h>
typedef enum SceFiosThreadType {
    SCE_FIOS_IO_THREAD = 0,
    SCE_FIOS_DECOMPRESSOR_THREAD = 1,
    SCE_FIOS_THREAD_TYPES = 2
} SceFiosThreadType;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_IO_THREAD	0	I/O thread.
SCE_FIOS_DECOMPRESSOR_THREAD	1	Decompressor thread.
SCE_FIOS_THREAD_TYPES	2	Total number of thread types.

Description

Thread types. These thread types are used as indices into SceFiosParams.threadPriority[].

See Also

SceFiosParams

SceFiosTime

An absolute point in time.

Definition

```
#include <fios2_types.h>
typedef int64_t SceFiosTime;
```

Description

An absolute point in time.

See Also

sceFiosTimeGetCurrent(), sceFiosTimeRelativeNanoseconds(), sceFiosTimeRelativeMicroseconds(), sceFiosTimeRelativeMilliseconds(), sceFiosTimeRelativeSeconds()

SceFiosTimeInterval

An interval between two points in time, using the same units as SceFiosTime.

Definition

```
#include <fios2_types.h>
typedef SceFiosTime SceFiosTimeInterval;
```

Description

An interval between two points in time, using the same units as SceFiosTime.

See Also

sceFiosTimeIntervalToNanoseconds(), sceFiosTimeIntervalFromNanoseconds(), sceFiosTimeIntervalToMicroseconds(), sceFiosTimeIntervalFromMicroseconds(), sceFiosTimeIntervalToMilliseconds(), sceFiosTimeIntervalFromMilliseconds(), sceFiosTimeIntervalToSeconds(), sceFiosTimeIntervalFromSeconds()

SceFiosTuple

A byte range inside a file.

Definition

```
#include <fios2_types.h>
typedef struct SceFiosTuple {
    SceFiosOffset offset;
    SceFiosSize size;
    char path[SCE_FIOS_PATH_MAX];
} SceFiosTuple;
```

Members

offset	Start of the byte range.
size	Length of the byte range, or SCE_FIOS_OFFSET_MAX.
path	Full path.

Description

A byte range inside a file.

See Also

sceFiosResolve(), sceFiosResolveSync()

SceFiosWhence

Whence values.

Definition

```
#include <fios2_types.h>
typedef enum SceFiosWhence {
    SCE_FIOS_SEEK_SET = 0,
    SCE_FIOS_SEEK_CUR = 1,
    SCE_FIOS_SEEK_END = 2
} SceFiosWhence;
```

Enumeration Values

Macro	Value	Description
SCE_FIOS_SEEK_SET	0	Absolute seek location from the beginning of file.
SCE_FIOS_SEEK_CUR	1	Relative seek location, based on current position.
SCE_FIOS_SEEK_END	2	Relative seek location, from the end of file.

Description

Whence values.

See Also

sceFiosFHSeek()