NAME

ost::AudioFile − A class used to manipulate audio data.

SYNOPSIS

`#include <audio2.h>`

Inherits **ost::AudioBase**.

Inherited by **ost::AudioStream**.

Public Member Functions

AudioFile (const char *name, unsigned long offset=0)
Construct and open an existing audio file for read/write.

AudioFile (const char *name, **Info** ***info**, unsigned long **minimum**=0)
Create and open a new audio file for writing.

AudioFile ()
Construct an audio file without attaching to the filesystem.

virtual **~AudioFile** ()

void **open** (const char *name, **Mode mode**=modeWrite, **timeout_t** framing=0)
Open an audio file and associate it with this object.

void **create** (const char *name, **Info** ***info**, bool exclusive=false, **timeout_t** framing=0)
Create a new audio file and associate it with this object.

time_t **getAge** (void)
Returns age since last prior access.

size_t **getSize** (void)
Get maximum size of frame buffer for data use.

void **close** (void)
Close an object associated with an open file.

void **clear** (void)
Clear the **AudioFile** *structure.*

ssize_t **getBuffer** (**Encoded** buffer, size_t len=0)
Retrieve bytes from the file into a memory buffer.

unsigned **getLinear** (**Linear** buffer, unsigned request=0)
Retrieve and convert content to linear encoded audio data from it's original form.

ssize_t **putBuffer** (**Encoded** buffer, size_t len=0)
Insert bytes into the file from a memory buffer.

unsigned **putLinear** (**Linear** buffer, unsigned request=0)
Convert and store content from linear encoded audio data to the format of the audio file.

Error getSamples (void *buffer, unsigned samples=0)
Retrieve samples from the file into a memory buffer.

Error putSamples (void *buffer, unsigned samples=0)
Insert samples into the file from a memory buffer.

Error skip (long number)
Change the file position by skipping a specified number of audio samples of audio data.

Error setPosition (unsigned long samples=~0l)
Seek a file position by sample count.

Error position (const char *timestamp)
Seek a file position by timestamp.

void **getPosition** (char *timestamp, size_t size)
Return the timestamp of the current absolute file position.

Error setLimit (unsigned long maximum=0l)
Set the maximum file position for reading and writing of audio data by samples.

Error getInfo (**Info** ***info**)
Copy the source description of the audio file into the specified object.

Error setMinimum (unsigned long **minimum**)
Set minimum file size for a created file.

unsigned long **getAbsolutePosition** (void)
Get the current file pointer in bytes relative to the start of the file.

unsigned long **getPosition** (void)
Get the current file pointer in samples relative to the start of the sample buffer.

virtual bool **isOpen** (void)
Test if the file is opened.
virtual bool **hasPositioning** (void)
Return true if underlying derived class supports direct access to file positioning.
Encoding getEncoding (void)
Return audio encoding format for this audio file.
Format getFormat (void)
Return base file format of containing audio file.
unsigned **getSampleRate** (void)
Get audio encoding sample rate, in samples per second, for this audio file.
char * **getAnnotation** (void)
Get annotation extracted from header of containing file.
Error getError (void)
Get last error code.
bool **operator!** (void)
bool **isSigned** (void)
Return if the current content is signed or unsigned samples.

Protected Member Functions

void **initialize** (void)
void **getWaveFormat** (int size)
void **mp3info** (**mpeg_audio** *mp3)
virtual bool **afCreate** (const char *path, bool exclusive=false)
virtual bool **afOpen** (const char *path, **Mode** m=modeWrite)
virtual bool **afPeek** (unsigned char *data, unsigned size)
AudioCodec * **getCodec** (void)
virtual int **afRead** (unsigned char *data, unsigned size)
Read a given number of bytes from the file, starting from the current file pointer.
virtual int **afWrite** (unsigned char *data, unsigned size)
Write a number of bytes into the file at the current file pointer.
virtual bool **afSeek** (unsigned long pos)
Seek to the given position relative to the start of the file and set the file pointer.
virtual void **afClose** (void)
Close the derived file handling system's file handle.
virtual char * **getContinuation** (void)
This function is used to splice multiple audio files together into a single stream of continues audio data.
const char * **getErrorStr** (**Error** err)
Return a human-readable error message given a numeric error code of type **Audio::Error**.
Error setError (**Error** err)
unsigned long **getHeader** (void)
Get number of bytes in the file header.
unsigned short **getShort** (unsigned char *data)
Convert binary 2 byte data stored in the order specified in the source description into a short variable.
void **setShort** (unsigned char *data, unsigned short value)
Save a short as two byte binary data stored in the endian order specified in the source description.
unsigned long **getLong** (unsigned char *data)
Convert binary 4 byte data stored in the order specified in the source description into a long variable.
void **setLong** (unsigned char *data, unsigned long value)
Save a long as four byte binary data stored in the endian order specified in the source description.

Protected Attributes

char * **pathname**
Error error
unsigned long **header**
unsigned long **minimum**

unsigned long **length**
union {
int **fd**
void * **handle**
} **file**
Mode mode
unsigned long **iolimit**

Detailed Description

A class used to manipulate audio data.

This class provides file level access to audio data stored in different formats. This class also provides the ability to write audio data into a disk file.

Author:
David Sugar <dyfet AT ostel DOT com> audio file access.

Constructor & Destructor Documentation

ost::AudioFile::AudioFile (const char * name, unsigned long offset = `0`**)**
Construct and open an existing audio file for read/write.

Parameters:
name of file to open.
offset to start access.

ost::AudioFile::AudioFile (const char * name, Info * info, unsigned long minimum = `0`**)**
Create and open a new audio file for writing.

Parameters:
name of file to create.
info source description for new file.
minimum file size to accept at close.

ost::AudioFile::AudioFile () `[inline]`
Construct an audio file without attaching to the filesystem.

virtual ost::AudioFile::~AudioFile () `[virtual]`

Member Function Documentation

void ost::AudioFile::initialize (void) `[protected]`
void ost::AudioFile::getWaveFormat (int size) `[protected]`
void ost::AudioFile::mp3info (mpeg_audio * mp3) `[protected]`
virtual bool ost::AudioFile::afCreate (const char * path, bool exclusive = `false`**)** `[protected, virtual]`
virtual bool ost::AudioFile::afOpen (const char * path, Mode m = `modeWrite`**)** `[protected, virtual]`
virtual bool ost::AudioFile::afPeek (unsigned char * data, unsigned size) `[protected, virtual]`
AudioCodec* ost::AudioFile::getCodec (void) `[protected]`
Reimplemented in **ost::AudioStream**.

virtual int ost::AudioFile::afRead (unsigned char * data, unsigned size) `[protected, virtual]`
Read a given number of bytes from the file, starting from the current file pointer.

May be overridden by derived classes.

Parameters:
data A pointer to the buffer to copy the bytes to.
size The number of bytes to read.

Returns:
The number of bytes read, or -1 if an error occurs. On UNIX platforms, use strerror(errno) to get the human-readable error string or FormatMessage(GetLastError()) on Windows platforms.

virtual int ost::AudioFile::afWrite (unsigned char * data, unsigned size) `[protected, virtual]`
Write a number of bytes into the file at the current file pointer.

May be overridden by derived classes.

Parameters:

data A pointer to the buffer with the bytes to write.
size The number of bytes to write from the buffer.

Returns:
The number of bytes written, or -1 if an error occurs. On UNIX platforms, use strerror(errno) to get the human-readable error string or FormatMessage(GetLastError()) on Windows platforms.

virtual bool ost::AudioFile::afSeek (unsigned long pos) `[protected, virtual]`
Seek to the given position relative to the start of the file and set the file pointer.

This does not use 64-bit clean seek functions, so seeking to positions greater than (2^32)-1 will result in undefined behavior.

Parameters:
pos The position to seek to.

Returns:
true if successful, false otherwise.

virtual void ost::AudioFile::afClose (void) `[protected, virtual]`
Close the derived file handling system's file handle.

virtual char* ost::AudioFile::getContinuation (void) `[inline, protected, virtual]`
This function is used to splice multiple audio files together into a single stream of continues audio data.

The continuation method returns the next audio file to open.

Returns:
next file to open or NULL when done.

const char* ost::AudioFile::getErrorStr (Error err) `[protected]`
Return a human-readable error message given a numeric error code of type **Audio::Error**.

Parameters:
err The numeric error code to translate.

Returns:
A pointer to a character string containing the human-readable error message.

Error ost::AudioFile::setError (Error err) `[protected]`
unsigned long ost::AudioFile::getHeader (void) `[inline, protected]`
Get number of bytes in the file header.

Data packets will begin after this header.

Returns:
number of bytes in file header.

unsigned short ost::AudioFile::getShort (unsigned char * data) `[protected]`
Convert binary 2 byte data stored in the order specified in the source description into a short variable.

This is often used to manipulate header data.

Returns:
short value.

Parameters:
data binary 2 byte data pointer.

void ost::AudioFile::setShort (unsigned char * data, unsigned short value) `[protected]`
Save a short as two byte binary data stored in the endian order specified in the source description.

This is often used to manipulate header data.

Parameters:
data binary 2 byte data pointer.
value to convert.

unsigned long ost::AudioFile::getLong (unsigned char * data) `[protected]`
Convert binary 4 byte data stored in the order specified in the source description into a long variable.

This is often used to manipulate header data.

Returns:

long value.

Parameters:
data binary 4 byte data pointer.

void ost::AudioFile::setLong (unsigned char * data, unsigned long value) `[protected]`
Save a long as four byte binary data stored in the endian order specified in the source description.

This is often used to manipulate header data.

Parameters:
data binary 4 byte data pointer.
value to convert.

void ost::AudioFile::open (const char * name, Mode mode = `modeWrite`**, timeout_t framing** = `0`**)**
Open an audio file and associate it with this object.

Called implicitly by the two-argument version of the constructor.

Parameters:
name of the file to open. Don't forget to double your backslashes for DOS-style pathnames.
mode to open file under.
framing time in milliseconds.

Reimplemented in **ost::AudioStream**.

void ost::AudioFile::create (const char * name, Info * info, bool exclusive = `false`**, timeout_t framing** = `0`**)**
Create a new audio file and associate it with this object.

Called implicitly by the three-argument version of the constructor.

Parameters:
name The name of the file to open.
info The type of the audio file to be created.
exclusive create option.
framing time in milliseconds.

Reimplemented in **ost::AudioStream**.

time_t ost::AudioFile::getAge (void)
Returns age since last prior access.

Used for cache computations.

Returns:
age in seconds.

size_t ost::AudioFile::getSize (void) `[inline]`
Get maximum size of frame buffer for data use.

Returns:
max frame size in bytes.

void ost::AudioFile::close (void)
Close an object associated with an open file.

This updates the header metadata with the file length if the file length has changed.

Reimplemented in **ost::AudioStream**.

void ost::AudioFile::clear (void)
Clear the **AudioFile** structure.

Called by **AudioFile::close()**. Sets all fields to zero and deletes the dynamically allocated memory pointed to by the pathname and info.annotation members. See **AudioFile::initialize()** for the dynamic allocation code.

ssize_t ost::AudioFile::getBuffer (Encoded buffer, size_t len = `0`**)** `[virtual]`
Retrieve bytes from the file into a memory buffer.

This increments the file pointer so subsequent calls read further bytes. If you want to read a number of samples rather than bytes, use **getSamples()**.

Parameters:
buffer area to copy the samples to.
len The number of bytes (not samples) to copy or 0 for frame.

Returns:
The number of bytes (not samples) read. Returns -1 if no bytes are read and an error occurs.

Implements **ost::AudioBase**.

Reimplemented in **ost::AudioStream**.

unsigned ost::AudioFile::getLinear (Linear buffer, unsigned request = `0`)
Retrieve and convert content to linear encoded audio data from it's original form.

Parameters:
buffer to copy linear data into.
request number of linear samples to extract or 0 for frame.

Returns:
number of samples retrieved, 0 if no codec or eof.

ssize_t ost::AudioFile::putBuffer (Encoded buffer, size_t len = `0`) `[virtual]`
Insert bytes into the file from a memory buffer.

This increments the file pointer so subsequent calls append further samples. If you want to write a number of samples rather than bytes, use **putSamples()**.

Parameters:
buffer area to append the samples from.
len The number of bytes (not samples) to append.

Returns:
The number of bytes (not samples) read. Returns -1 if an error occurs and no bytes are written.

Implements **ost::AudioBase**.

unsigned ost::AudioFile::putLinear (Linear buffer, unsigned request = `0`)
Convert and store content from linear encoded audio data to the format of the audio file.

Parameters:
buffer to copy linear data from.
request Number of linear samples to save or 0 for frame.

Returns:
number of samples saved, 0 if no codec or eof.

Error ost::AudioFile::getSamples (void * buffer, unsigned samples = `0`)
Retrieve samples from the file into a memory buffer.

This increments the file pointer so subsequent calls read further samples. If a limit has been set using **setLimit()**, the number of samples read will be truncated to the limit position. If you want to read a certain number of bytes rather than a certain number of samples, use **getBuffer()**.

Parameters:
buffer pointer to copy the samples to.
samples The number of samples to read or 0 for frame.

Returns:
errSuccess if successful, !errSuccess if error. Use **getErrorStr()** to retrieve the human-readable error string.

Error ost::AudioFile::putSamples (void * buffer, unsigned samples = `0`)
Insert samples into the file from a memory buffer.

This increments the file pointer so subsequent calls append further samples. If you want to write a certain number of bytes rather than a certain number of samples, use **putBuffer()**.

Parameters:
buffer pointer to append the samples from.
samples The number of samples (not bytes) to append.

Returns:

errSuccess if successful, !errSuccess if error. Use **getErrorStr()** to retrieve the human-readable error string.

Error ost::AudioFile::skip (long number)

Change the file position by skipping a specified number of audio samples of audio data.

Returns:

errSuccess or error condition on failure.

Parameters:

number of samples to skip.

Error ost::AudioFile::setPosition (unsigned long samples = `~0l`)

Seek a file position by sample count.

If no position specified, then seeks to end of file.

Returns:

errSuccess or error condition on failure.

Parameters:

samples position to seek in file.

Error ost::AudioFile::position (const char * timestamp)

Seek a file position by timestamp.

The actual position will be rounded by framing.

Returns:

errSuccess if successful.

Parameters:

timestamp position to seek.

void ost::AudioFile::getPosition (char * timestamp, size_t size)

Return the timestamp of the current absolute file position.

Parameters:

timestamp to save ascii position into.
size of timestamp buffer.

Error ost::AudioFile::setLimit (unsigned long maximum = `0l`)

Set the maximum file position for reading and writing of audio data by samples.

If 0, then no limit is set.

Parameters:

maximum file i/o access size sample position.

Returns:

errSuccess if successful.

Error ost::AudioFile::getInfo (Info * info)

Copy the source description of the audio file into the specified object.

Parameters:

info pointer to object to copy source description into.

Returns:

errSucess.

Error ost::AudioFile::setMinimum (unsigned long minimum)

Set minimum file size for a created file.

If the file is closed with fewer samples than this, it will also be deleted.

Parameters:

minimum number of samples for new file.

Returns:

errSuccess if successful.

unsigned long ost::AudioFile::getAbsolutePosition (void)
Get the current file pointer in bytes relative to the start of the file.

See **getPosition()** to determine the position relative to the start of the sample buffer.

Returns:
The current file pointer in bytes relative to the start of the file. Returns 0 if the file is not open, is empty, or an error has occured.

unsigned long ost::AudioFile::getPosition (void)
Get the current file pointer in samples relative to the start of the sample buffer.

Note that you must multiply this result by the result of a call to toBytes(info.encoding, 1) in order to determine the offset in bytes.

Returns:
the current file pointer in samples relative to the start of the sample buffer. Returns 0 if the file is not open, is empty, or an error has occured.

virtual bool ost::AudioFile::isOpen (void) `[virtual]`
Test if the file is opened.

Returns:
true if a file is open.

virtual bool ost::AudioFile::hasPositioning (void) `[inline, virtual]`
Return true if underlying derived class supports direct access to file positioning.

Derived classes based on URL's or fifo devices may not have this ability.

Returns:
true if file positioning is supported.

Encoding ost::AudioFile::getEncoding (void) `[inline]`
Return audio encoding format for this audio file.

Returns:
audio encoding format.

Reimplemented from **ost::AudioBase**.

Format ost::AudioFile::getFormat (void) `[inline]`
Return base file format of containing audio file.

Returns:
audio file container format.

unsigned ost::AudioFile::getSampleRate (void) `[inline]`
Get audio encoding sample rate, in samples per second, for this audio file.

Returns:
sample rate.

Reimplemented from **ost::AudioBase**.

char* ost::AudioFile::getAnnotation (void) `[inline]`
Get annotation extracted from header of containing file.

Returns:
annotation text if any, else NULL.

Error ost::AudioFile::getError (void) `[inline]`
Get last error code.

Returns:
alst error code.

bool ost::AudioFile::operator! (void) `[inline]`
bool ost::AudioFile::isSigned (void)
Return if the current content is signed or unsigned samples.

Returns:
true if signed.

Member Data Documentation

char* ost::AudioFile::pathname [protected]

Error ost::AudioFile::error [protected]

unsigned long ost::AudioFile::header [protected]

unsigned long ost::AudioFile::minimum [protected]

unsigned long ost::AudioFile::length [protected]

int ost::AudioFile::fd

void* ost::AudioFile::handle

union { ... } ost::AudioFile::file [protected]

Mode ost::AudioFile::mode [protected]

unsigned long ost::AudioFile::iolimit [protected]

Author

Generated automatically by Doxygen for ccAudio from the source code.