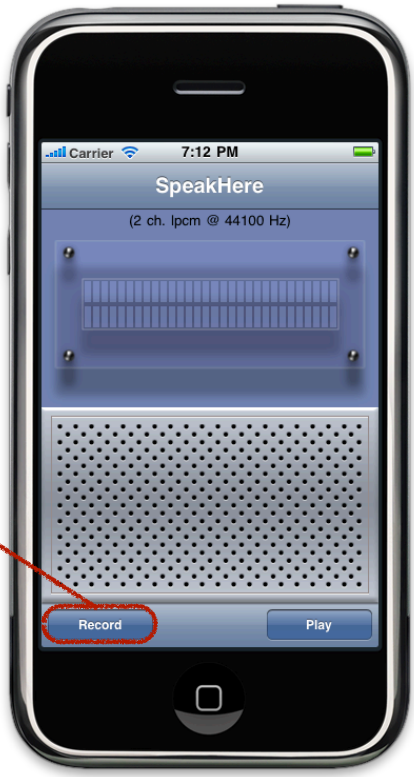
Playing & Recording Audio
Audio Queue Services
Recording - AQRecorder Class

사운드 콘텐츠 응용 10주차

2010-2
멀티미디어과학과 이종우

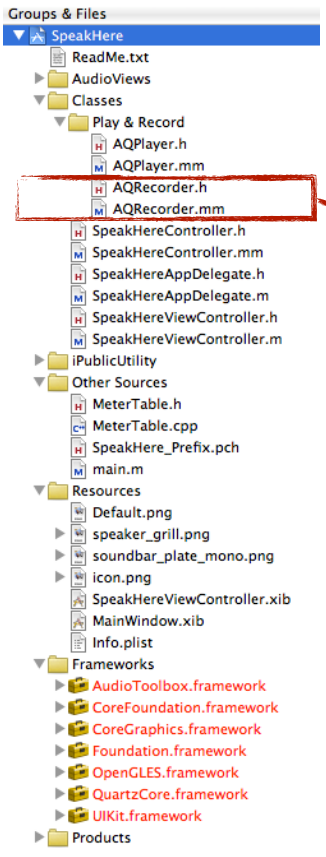
Index

AQRecorder



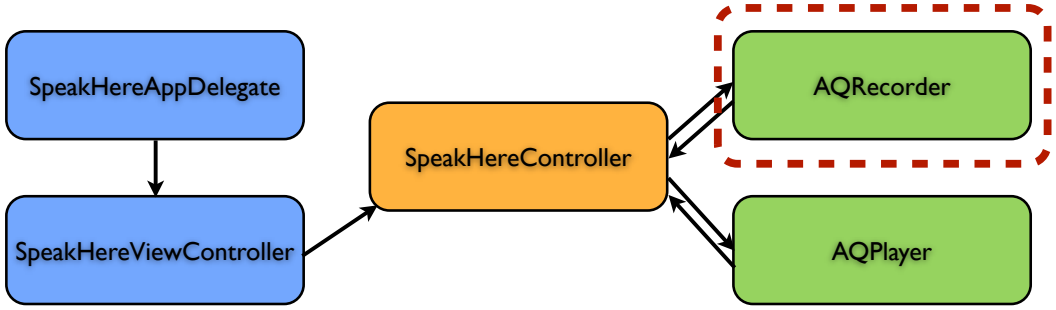
Sound Recoding

SpeakHere Classes & Resources



AQRecorder 클래스 : 레코더 객체를 위한 클래스

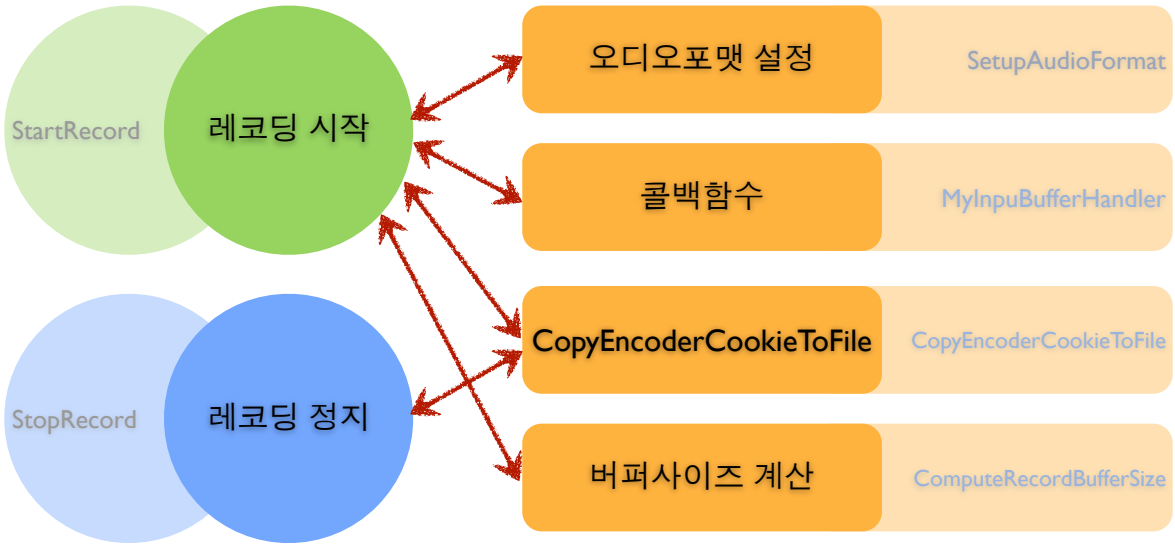
SpeakHere 클래스 관계도



이번 시간에는...
AQRecorder 클래스에서
녹음시작, 녹음정지를 각각 순서대로 메소드 흐름 분석!

AQRecorder Class

- AQRecorder 클래스는 6개의 메소드로 구성



AQRecorder.h

- AQRecorder 정의

1. public 인스턴스 & 메소드

```
AQRecorder();
~AQRecorder();

UInt32                  GetNumberChannels() const     { return mRecordFormat.NumberChannels(); }
CFStringRef             GetFileName() const           { return mFileName; }
AudioQueueRef           Queue() const                 { return mQueue; }
CAStreamBasicDescription DataFormat() const           { return mRecordFormat; }

void                    StartRecord(CFStringRef inRecordFile);
void                    StopRecord();
Boolean                 IsRunning() const             { return mIsRunning; }

UInt64              startTime;
```

2. private 인스턴스 & 메소드

```
CFStringRef                     mFileName;
AudioQueueRef                   mQueue;
AudioQueueBufferRef             mBuffers[kNumberRecordBuffers];
AudioFileID                     mRecordFile;
SInt64                          mRecordPacket; // current packet number in record file
CAStreamBasicDescription mRecordFormat;
Boolean                         mIsRunning;

void                CopyEncoderCookieToFile();
void                SetupAudioFormat(UInt32 inFormatID);
int                 ComputeRecordBufferSize(const AudioStreamBasicDescription *format, float seconds);

static void MyInputBufferHandler(  void *                                  inUserData,
                                   AudioQueueRef                           inAQ,
                                   AudioQueueBufferRef                     inBuffer,
                                   const AudioTimeStamp *                  inStartTime,
                                   UInt32                                  inNumPackets,
                                   const AudioStreamPacketDescription*     inPacketDesc);
```

Start Recording Flow

1. StartRecord 메소드 호출

- SpeakHereController로부터 AQRecorder::StartRecord 메소드 호출

1. AQRecorder 클래스 생성자
   - main() 함수 실행 시 mIsRunning=false 상태로 AQRecorder 생성자가 실행

```
AQRecorder::AQRecorder() {
                            mIsRunning = false;
                            mRecordPacket = 0;
}
```

2. 어플리케이션 화면에서 'Record' 버튼 선택 및 SpeakHereController의 IBAction record 메소드 호출

```
- (IBAction)record:(id)sender
{
   ...
}
```

SpeakHereController

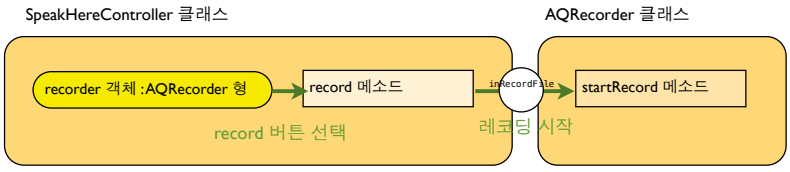
3. AQRecorder 클래스의 StartRecord 메소드 호출

```
recorder->StartRecord(CFSTR("recordedFile.caf"));
```

SpeakHereController

1. StartRecord 메소드 호출

- 현재까지 메소드 흐름도



2. 파일명 설정

1. 전달받은 인자로 레코딩 될 파일명 설정

- AQRecorder::**StartRecord** 메소드

```
void AQRecorder::StartRecord(CFStringRef inRecordFile)
{
    ...
try {
        mFileName = CFStringCreateCopy(kCFAllocatorDefault, inRecordFile);

        ...
...
```

```
void AQRecorder::StartRecord()
```

CFStringCreateCopy

Creates an immutable copy of a string.

```
CFStringRef CFStringCreateCopy (
   CFAllocatorRef alloc,
   CFStringRef theString
);
```

alloc	The allocator to use to allocate memory for the new string. Pass NULL or kCFAllocatorDefault to use the current default allocator.
theString	The string to copy.

3. 레코딩 포맷 설정

1. SetupAudioFormat 메소드 호출을 통해 레코딩 포맷 설정

• AQRecorder::**StartRecord** 메소드

```
// specify the recording format
SetupAudioFormat(kAudioFormatLinearPCM);
```

2. 전달받은 인자를 통해 메모리 및 오디오세션 속성을 설정

• AQRecorder::**SetupAudioFormat** 메소드

```
memset(&mRecordFormat, 0, sizeof(mRecordFormat));

UInt32 size = sizeof(mRecordFormat.mSampleRate);
XThrowIfError(AudioSessionGetProperty(kAudioSessionProperty_CurrentHardwareSampleRate, &size,
                                      &mRecordFormat.mSampleRate), "couldn't get hardware sample rate");

size = sizeof(mRecordFormat.mChannelsPerFrame);
XThrowIfError(AudioSessionGetProperty(kAudioSessionProperty_CurrentHardwareInputNumberChannels, &size,
                                      &mRecordFormat.mChannelsPerFrame), "couldn't get input channel count");
```

```
void AQRecorder::StartRecord()
```

`kAudioFormatLinearPCM`

A key that specifies linear PCM, a noncompressed audio data format with one frame per packet.

`memset(&mRecordFormat, 0, sizeof(mRecordFormat));`

메모리를 다루는 함수(C, C++)

memset(초기화할배열명, 초기화할배열값([int], 초기화할배염범위 [ex : 10 * sizeof(int)]);

```
void AQRecorder::SetupAudioFormat(UInt32 inFormatID)
```

AudioSessionGetProperty

Gets the value of a specified audio session property.

```
OSStatus AudioSessionGetProperty (
   AudioSessionPropertyID    inID,
   UInt32                    *ioDataSize,
   void                      *outData
);
```

inID	The identifier for the audio session property that you want to get the value of.
ioDataSize	On input, the memory size for the outData parameter. On output, the actual size of the property value.
outData	On output, the value of the specified audio session property.

3. 레코딩 포맷 설정

3. mRecordFormat 설정

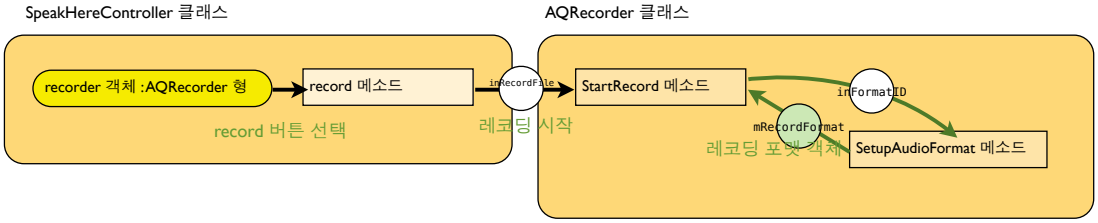
- CAStreamBasicDescription 형의 mRecordFormat
- mRecordFormat의 채널 비트 수, 패킷 수, 패킷 프레임 수 등 설정

• AQRecorder::**SetupAudioFormat** 메소드

```
mRecordFormat.mFormatID = inFormatID;
if (inFormatID == kAudioFormatLinearPCM)
{
    // if we want pcm, default to signed 16-bit little-endian
    mRecordFormat.mFormatFlags = kLinearPCMFormatFlagIsSignedInteger |
                                                        kLinearPCMFormatFlagIsPacked;
    mRecordFormat.mBitsPerChannel = 16;
    mRecordFormat.mBytesPerPacket = mRecordFormat.mBytesPerFrame =
                        (mRecordFormat.mBitsPerChannel / 8) * mRecordFormat.mChannelsPerFrame;
    mRecordFormat.mFramesPerPacket = 1;
```

3. 레코딩 포맷 설정

- 현재까지 메소드 흐름도



4. Queue 생성

1. MyInputBufferHandler 메소드를 참조하여 레코딩 파일을 넣을 Queue 생성

- AQRecorder::**StartRecord** 메소드

```
// create the queue
XThrowIfError(AudioQueueNewInput(
                              &mRecordFormat,
                              MyInputBufferHandler,
                              this /* userData */,
                              NULL /* run loop */, NULL /* run loop mode */,
                              0 /* flags */, &mQueue), "AudioQueueNewInput failed");
```

2. input 버퍼가 다 찼을 때 호출되는 오디오 Queue 콜백함수 동작

- AQRecorder::**MyInputBufferHandler** 메소드

```
void AQRecorder::MyInputBufferHandler(  void *                                  inUserData,
                                        AudioQueueRef                           inAQ,
                                        AudioQueueBufferRef                     inBuffer,
                                        const AudioTimeStamp *                  inStartTime,
                                        UInt32                                  inNumPackets,
                                        const AudioStreamPacketDescription*     inPacketDesc)
```

4. Queue 생성

3. input 버퍼가 다 찼을 때 호출되는 오디오 Queue 콜백함수 동작

- AQRecorder::**MyInputBufferHandler** 메소드

```
if (inNumPackets > 0) {
    // write packets to file
    XThrowIfError(AudioFileWritePackets(aqr->mRecordFile, FALSE, inBuffer->mAudioDataByteSize,
                  inPacketDesc, aqr->mRecordPacket, &inNumPackets, inBuffer->mAudioData),
                  "AudioFileWritePackets failed");
    aqr->mRecordPacket += inNumPackets;
}

// if we're not stopping, re-enqueue this buffer so that it gets filled again
if (aqr->IsRunning())
    XThrowIfError(AudioQueueEnqueueBuffer(inAQ, inBuffer, 0, NULL), "AudioQueueEnqueueBuffer failed");
```

```
void AQRecorder::StartRecord()
```

AudioQueueNewInput

Creates a new recording audio queue object.

```
OSStatus AudioQueueNewInput (
   const AudioStreamBasicDescription  *inFormat,
   AudioQueueInputCallback            inCallbackProc,
   void                               *inUserData,
   CFRunLoopRef                       inCallbackRunLoop,
   CFStringRef                        inCallbackRunLoopMode,
   UInt32                             inFlags,
   AudioQueueRef                      * inFlags
);
```

inFormat	The compressed or uncompressed audio data format to record to.
inCallbackProc	A callback function to use with the recording audio queue.
inUserData	A custom data structure for use with the callback function.
inCallbackRunLoop	The event loop on which the callback function pointed to by the inCallbackProc parameter
inCallbackRunLoopMode	The run loop mode in which to invoke the callback function specified in the inCallbackProc parameter.
inFlags	Reserved for future use. Must be 0.
inFlags	On output, the newly created recording audio queue.

```
void AQRecorder::MyInputBufferHandler()
```

AudioQueueInputCallback

Called by the system when a recording audio queue has finished filling an audio queue buffer.

```
void AQRecorder::MyInputBufferHandler(    void *                            inUserData,
                                          AudioQueueRef                     inAQ,
                                          AudioQueueBufferRef               inBuffer,
                                          const AudioTimeStamp *            inStartTime,
                                          UInt32                            inNumPackets,
                                          const AudioStreamPacketDescription* inPacketDesc)
```

`inUserData`	The custom data you've specified in the `inUserData` parameter of the `AudioQueueNewInput` function. Typically, this includes format and state information for the audio queue.
`inAQ`	The recording audio queue that invoked the callback.
`inBuffer`	An audio queue buffer, newly filled by the recording audio queue, containing the new audio data your callback needs to write.
`inStartTime`	The sample time for the start of the audio queue buffer. This parameter is not used in basic recording.
`inNumberPacketDescriptions`	The number of packets of audio data sent to the callback in the inBuffer parameter. When recording in a constant bit rate (CBR) format, the audio queue sets this parameter to `NULL`.
`inPacketDescs`	For compressed formats that require packet descriptions, the set of packet descriptions produced by the encoder for audio data in the inBuffer parameter. When recording in a CBR format, the audio queue sets this parameter to `NULL`.


```
void AQRecorder::MyInputBufferHandler()
```


AudioFileWritePackets

Writes packets of audio data to an audio data file.

```
OSStatus AudioFileWritePackets (
   AudioFileID                     inAudioFile,
   Boolean                         inUseCache,
   UInt32                          inNumBytes,
   const AudioStreamPacketDescription *inPacketDescriptions,
   SInt64                          inStartingPacket,
   UInt32                          *ioNumPackets,
   const void                      *inBuffer
);
```

`inAudioFile`	The audio file to write to.
`inUseCache`	Set to `true` if you want to cache the data. Otherwise, set to `false`.
`inNumBytes`	The number of bytes of audio data being written.
`inPacketDescriptions`	A pointer to an array of packet descriptions for the audio data. Not all formats require packet descriptions. If no packet descriptions are required, for instance, if you are writing CBR data, pass `NULL`.
`inStartingPacket`	The packet index for the placement of the first provided packet.
`ioNumPackets`	On input, a pointer to the number of packets to write. On output, a pointer to the number of packets actually written.
`inBuffer`	A pointer to user-allocated memory containing the new audio data to write to the audio data file.

```
void AQRecorder::MyInputBufferHandler()
```

AudioQueueEnqueueBuffer

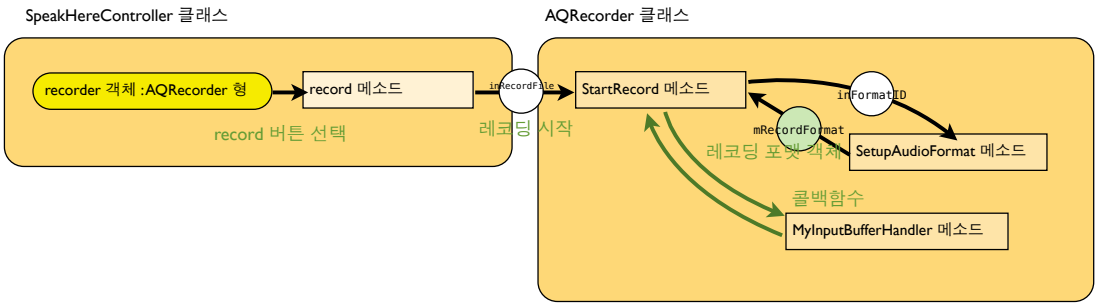
Adds a buffer to the buffer queue of a recording or playback audio queue.

```
OSStatus AudioQueueEnqueueBuffer (
   AudioQueueRef                     inAQ,
   AudioQueueBufferRef               inBuffer,
   UInt32                            inNumPacketDescs,
   const AudioStreamPacketDescription *inPacketDescs
);
```

`inAQ`	The audio queue that owns the audio queue buffer.
`inBuffer`	The audio queue buffer to add to the buffer queue.
`inNumPacketDescs`	The number of packets of audio data in the inBuffer parameter. Use a value of `0` for any of the following situations: When playing a constant bit rate (CBR) format. When the audio queue is a recording (input) audio queue. When the buffer you are reenqueuing was allocated with the function.
`inPacketDescs`	An array of packet descriptions. Use a value of `NULL` for any of the following situations: When playing a constant bit rate (CBR) format. When the audio queue is an input (recording) audio queue. When the buffer you are reenqueuing was allocated with the `AudioQueueAllocateBufferWithPacketDescriptions` function.

4. Queue 생성

- 현재까지 메소드 흐름도



5. 레코딩 파일 URL 설정

1. 레코딩 파일의 URL 설정

- AQRecorder::**StartRecord** 메소드

```
// get the record format back from the queue's audio converter --
// the file may require a more specific stream description than was necessary to create the encoder.
mRecordPacket = 0;

size = sizeof(mRecordFormat);
XThrowIfError(AudioQueueGetProperty(mQueue, kAudioQueueProperty_StreamDescription,
                                    &mRecordFormat, &size), "couldn't get queue's format");

NSString *recordFile = [NSTemporaryDirectory()
                        stringByAppendingPathComponent: (NSString*)inRecordFile];

url = CFURLCreateWithString(kCFAllocatorDefault, (CFStringRef)recordFile, NULL);
```

```
void AQRecorder::StartRecord()
```

AudioQueueGetProperty

Gets an audio queue property value.

```
OSStatus AudioQueueGetProperty (
    AudioQueueRef inAQ,
    AudioQueuePropertyID inID,
    void *outData,
    UInt32 *ioDataSize
);
```

inAQ	The audio queue that you want to get a property value from.
inID	The ID of the property whose value you want to get.
outData	On output, the desired property value.
ioDataSize	On input, the maximum bytes of space the caller expects to receive. On output, the actual data size of the property value.

```
void AQRecorder::StartRecord()
```

NSTemporaryDirectory

Returns the path of the temporary directory for the current user.

```
NSString * NSTemporaryDirectory (void);
```

stringByAppendingPathComponent

Returns a new string made by appending to the receiver a given string.

```
- (NSString *)stringByAppendingPathComponent:(NSString *)aString
```

aString	The path component to append to the receiver.


```
void AQRecorder::StartRecord()
```


CFURLCreateWithString

Creates a `CFURL` object using a given `CFString` object.

```
CFURLRef CFURLCreateWithString (
   CFAllocatorRef allocator,
   CFStringRef URLString,
   CFURLRef baseURL
);
```

`allocator`	The allocator to use to allocate memory for the new `CFURL` object. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.
`URLString`	The `CFString` object containing the URL string.
`baseURL`	The URL to which `URLString` is relative. Pass `NULL` if URLString contains an absolute URL or if you want to create a relative URL. If `URLString` contains an absolute URL, `baseURL` is ignored.

6. 오디오파일 생성

1. 레코딩 된 오디오 파일 생성

- AQRecorder::**StartRecord** 메소드

```
// create the audio file
XThrowIfError(AudioFileCreateWithURL(url, kAudioFileCAFType, &mRecordFormat,
                                     kAudioFileFlags_EraseFile, &mRecordFile),
                                     "AudioFileCreateWithURL failed");

CFRelease(url);
```

```
void AQRecorder::StartRecord()
```

AudioFileCreateWithURL

Creates a new audio file, or initializes an existing file, specified by a URL.

```
OSStatus AudioFileCreateWithURL (
   CFURLRef                         inFileRef,
   AudioFileTypeID                  inFileType,
   const AudioStreamBasicDescription *inFormat,
   UInt32                           inFlags,
   AudioFileID                      *outAudioFile
);
```

inFileRef	The fully specified path of the file to create or initialize.
inFileType	The type of audio file to create.
inFormat	A pointer to the structure that describes the format of the data.
inFlags	Relevant flags for creating or opening the file. If kAudioFileFlags_EraseFile is set, it erases an existing file. If the flag is not set, the function fails fails if the URL is an existing file.
outAudioFile	On output, a pointer to a newly created or initialized file.

```
void AQRecorder::StartRecord()
```

CFRelease

Releases a Core Foundation object.

```
void CFRelease (
   CFTypeRef cf
);
```

cf	A CFType object to release. This value must not be NULL.

7. Magic Cookie 설정

1. CopyEncoderCookieToFile 메소드 호출을 통해

- AQRecorder::**StartRecord** 메소드

```
// copy the cookie first to give the file object as much info as we can about the data going in
// not necessary for pcm, but required for some compressed audio
CopyEncoderCookieToFile();
```

2. 오디오 파일에 큐 인코더의 마법쿠키를 복사하는 메소드

- AQRecorder::**CopyEncoderCookieToFile** 메소드

```
OSStatus err = AudioQueueGetPropertySize(mQueue, kAudioQueueProperty_MagicCookie, &propertySize);
...
err = AudioFileGetPropertyInfo(mRecordFile, kAudioFilePropertyMagicCookieData, NULL, &willEatTheCookie);

if (err == noErr && willEatTheCookie) {

    err = AudioFileSetProperty(mRecordFile, kAudioFilePropertyMagicCookieData, magicCookieSize, magicCookie);
    XThrowIfError(err, "set audio file's magic cookie");
...
```

```
void AQRecorder::CopyEncoderCookieToFile()
```

AudioQueueGetPropertySize

Gets the size of the value of an audio queue property.

```
OSStatus AudioQueueGetPropertySize (
   AudioQueueRef inAQ,
   AudioQueuePropertyID inID,
   UInt32 *outDataSize
);
```

inAQ	The audio queue that has the property value whose size you want to get.
inID	The ID of the property value whose size you want to get
outDataSize	On output, the size of the requested property value.


```
void AQRecorder::CopyEncoderCookieToFile()
```


AudioFileGetPropertyInfo

Gets information about an audio file property, including the size of the property value and whether the value is writable.

```
OSStatus AudioFileGetPropertyInfo (
   AudioFileID         inAudioFile,
   AudioFilePropertyID inPropertyID,
   UInt32              *outDataSize,
   UInt32              *isWritable
);
```

inAudioFile	The audio file you want to obtain property value information from.
inPropertyID	The property whose value information you want.
outDataSize	On output, the size in bytes of the property value.
isWritable	On output, equals `1` if the property is writable, or `0` if it is read-only.

```
void AQRecorder::CopyEncoderCookieToFile()
```

AudioFileSetProperty

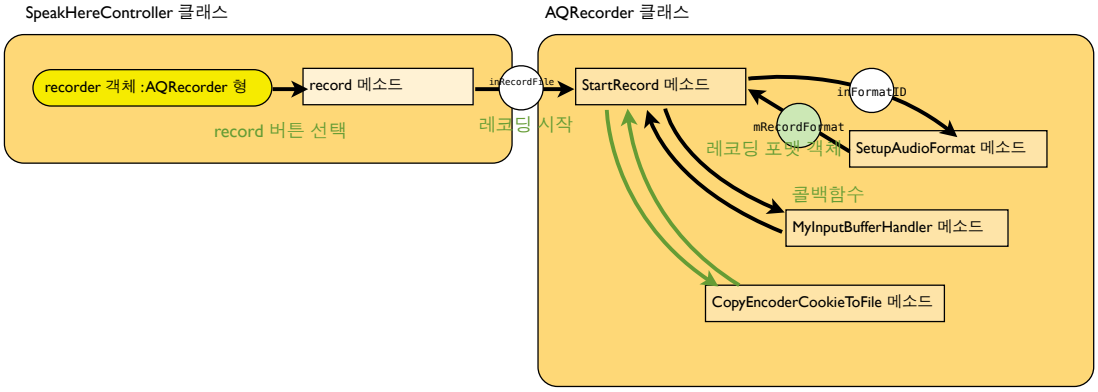
Sets the value of an audio file property

```
OSStatus AudioFileSetProperty (
   AudioFileID         inAudioFile,
   AudioFilePropertyID inPropertyID,
   UInt32              inDataSize,
   const void          *inPropertyData
);
```

`inAudioFile`	The audio file that you want to set a property value for.
`inPropertyID`	The property whose value you want to set.
`inDataSize`	The size of the value you are passing in the inPropertyData parameter.
`inPropertyData`	The new value for the property.

7. Magic Cookie 설정

- 현재까지 메소드 흐름도



8. 버퍼를 할당하고 큐에 채움

1. AudioQueueAllocateBuffer 메소드 호출을 통해 버퍼 할당 & Enqueue

- AQRecorder::**StartRecord** 메소드

```
// allocate and enqueue buffers
bufferByteSize = ComputeRecordBufferSize(&mRecordFormat, kBufferDurationSeconds);
                                                    // enough bytes for half a second
for (i = 0; i < kNumberRecordBuffers; ++i) {
          XThrowIfError(AudioQueueAllocateBuffer(mQueue, bufferByteSize, &mBuffers[i]),
                        "AudioQueueAllocateBuffer failed");
          XThrowIfError(AudioQueueEnqueueBuffer(mQueue, mBuffers[i], 0, NULL),
                        "AudioQueueEnqueueBuffer failed");
}
```

2. 버퍼 사이즈 계산

- AQRecorder::**ComputeRecordBufferSize** 메소드

```
frames = (int)ceil(seconds * format->mSampleRate);

if (format->mBytesPerFrame > 0)
    bytes = frames * format->mBytesPerFrame;
else {
    UInt32 maxPacketSize;
    if (format->mBytesPerPacket > 0)
          maxPacketSize = format->mBytesPerPacket;     // constant packet size
    else {
          UInt32 propertySize = sizeof(maxPacketSize);
          XThrowIfError(AudioQueueGetProperty(mQueue, kAudioQueueProperty_MaximumOutputPacketSize,
                        &maxPacketSize, &propertySize), "couldn't get queue's maximum output packet size");
    }
    if (format->mFramesPerPacket > 0)
          packets = frames / format->mFramesPerPacket;
    else
          packets = frames;   // worst-case scenario: 1 frame in a packet
    if (packets == 0)         // sanity check
          packets = 1;
    bytes = packets * maxPacketSize;
}
```

```
void AQRecorder::StartRecord()
```

AudioQueueAllocateBuffer

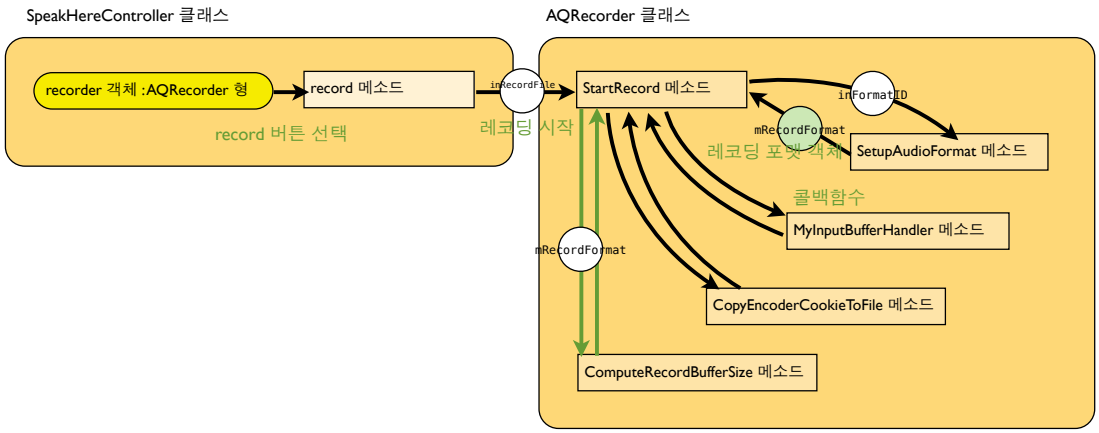
Asks an audio queue object to allocate an audio queue buffer.

```
OSStatus AudioQueueAllocateBuffer (
   AudioQueueRef inAQ,
   UInt32 inBufferByteSize,
   AudioQueueBufferRef *outBuffer
);
```

inAQ	The audio queue you want to allocate a buffer.
inBufferByteSize	The desired capacity of the new buffer, in bytes.
outBuffer	On output, points to the newly allocated audio queue buffer.

8. 버퍼를 할당하고 큐에 채움

- 현재까지 메소드 흐름도



9. Queue 시작

1. mIsRunning 상태 설정

- AQRecorder::**StartRecord** 메소드

```
// start the queue
mIsRunning = true;
```

2. Queue 시작

- AQRecorder::**StartRecord** 메소드

```
// start the queue
XThrowIfError(AudioQueueStart(mQueue, NULL), "AudioQueueStart failed");
```

```
void AQRecorder::StartRecord()
```

AudioQueueStart

Begins playing or recording audio.

```
OSStatus AudioQueueStart (
   AudioQueueRef inAQ,
   const AudioTimeStamp *inStartTime
);
```

inAQ	The audio queue to start.
inStartTime	The time at which the audio queue should start.

Stop Recording Flow

1. StopRecord 메소드 호출

- SpeakHereController로부터 AQRecorder::StopRecord 메소드 호출

1. 어플리케이션 화면에서 'Stop' 버튼 선택 및 SpeakHereController의 IBAction stopRecord 메소드 호출
   - SpeakHereController::**stopRecord** 메소드

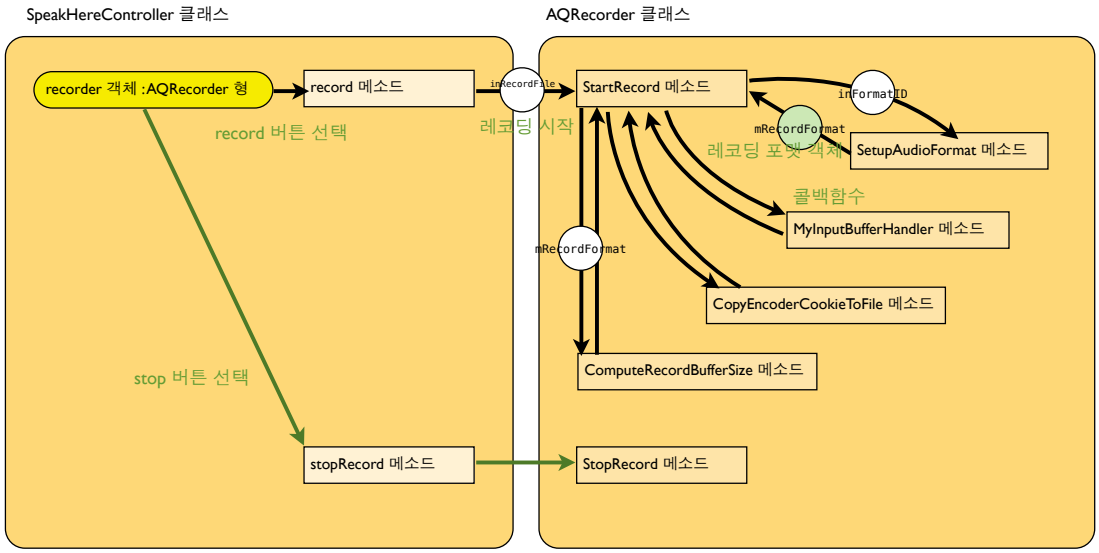
```
- (void)stopRecord
{
   ...
}
```

2. AQRecorder 클래스의 StopRecord 메소드 호출
   - SpeakHereController::**stopRecord** 메소드

```
recorder->StopRecord();
```

1. StopRecord 메소드 호출

- **startRecord** 메소드 흐름도



2. 레코딩 종료

1. mIsRunning 상태 설정

- SpeakHereController::**stopRecord** 메소드

```
// end recording
mIsRunning = false;
```

2. AQRecorder 클래스의 StopRecord 메소드 호출

- SpeakHereController::**stopRecord** 메소드

```
// end recording

XThrowIfError(AudioQueueStop(mQueue, true), "AudioQueueStop failed");
```

```
void AQRecorder::StopRecord()
```

AudioQueueStop

Stops playing or recording audio.

```
OSStatus AudioQueueStop (
   AudioQueueRef inAQ,
   Boolean inImmediate
);
```

inAQ	The audio queue to stop.
inImmediate	If you pass true, stopping occurs immediately (that is, synchronously). If you pass false, the function returns immediately, but the audio queue does not stop until its queued buffers are played or recorded (that is, the stop occurs asynchronously).

3. Magic Cookie 업데이트

1. 종료되었으므로 Magic Cookie를 업데이트함

- SpeakHereController::**stopRecord** 메소드

```
// a codec may update its cookie at the end of an encoding session, so reapply it to the file now
CopyEncoderCookieToFile();
```

4. 파일명 공간 해제

1. mFileName release 및 NULL 설정

- SpeakHereController::**stopRecord** 메소드

```
if (mFileName)
{
     CFRelease(mFileName);
     mFileName = NULL;
}
```

5. 오디오 큐 해제 및 종료

1. 오디오 큐 해제 및 파일 닫기

- SpeakHereController::**stopRecord** 메소드

```
AudioQueueDispose(mQueue, true);

AudioFileClose(mRecordFile);
```

```
void AQRecorder::StopRecord()
```

AudioQueueDispose

Disposes of an audio queue.

```
OSStatus AudioQueueDispose (
   AudioQueueRef inAQ,
   Boolean inImmediate
);
```

`inAQ`	The audio queue you want to dispose of.
`inImmediate`	If you pass `true`, the audio queue is disposed of immediately (that is, synchronously). If you pass `false`, disposal does not take place until all enqueued buffers are processed (that is, asynchronously).

```
void AQRecorder::StopRecord()
```

AudioFileClose

Closes an audio file.

```
OSStatus AudioFileClose (
   AudioFileID inAudioFile
);
```

inAudioFile	The file you want to close.

next time...