



CAudioRecorder illustrates how to use CMdaAudioRecorderUtility for recording wav files.

CAudioRecorder requires MMdaObjectStateChangeObserver callback interface to be implemented by the calling class. The only function defined in this interface is used to update different states of the file playing. The recording to the file can start only after this interface function is called with EOpen. Note that EOpen is also given when the audio recording finishes, thus you should also check the previous state variable before calling any functions.

This example also requires calling class to implement own callback interface function that is used for updating the playing status. This status is updated according to the players status as well the playing process is updated using timer implementation.

Headers:

```
#include <MdaAudioSampleEditor.h>
#include <Mda\Client\Utility.h>
#include <Mda\Common\Resource.h>
```

Link against:

```
LIBRARY mediaclientaudio.lib
```

Capabilities:

```
CAPABILITY UserEnvironment
```

AudioRecorder.cpp

```
#include <e32base.h>
#include <aknviewappui.h>

#include <Mda\Common\Resource.h>
#include <BAUTILS.H>

#include "AudioRecorder.h"

const TInt KRefreshTimeOut = 1000000; // re-fresh every second

CAudioRecorder* CAudioRecorder::NewL(MExmapleRecStateObserver& aObserver)
{
    CAudioRecorder* self = CAudioRecorder::NewLC(aObserver);
    CleanupStack::Pop(self);
    return self;
}

CAudioRecorder* CAudioRecorder::NewLC(MExmapleRecStateObserver& aObserver)
{
    CAudioRecorder* self = new (ELeave) CAudioRecorder(aObserver);
    CleanupStack::PushL(self);
    self->ConstructL();
    return self;
}
```

Record_audio_files

```
}

CAudioRecorder::CAudioRecorder(MExampleRecStateObserver& aObserver)
:iObserver(aObserver), iVolume(5)
{
}

CAudioRecorder::~CAudioRecorder()
{
    delete iExampleTimer;
    if(iToneUtility)
    {
        iToneUtility->Stop();
        iToneUtility->Close();
    }

    delete iToneUtility;
    delete iFormat;
    delete iCodec;
    delete iSettings;
}

void CAudioRecorder::ConstructL()
{
    iExampleTimer = CExampleTimer::NewL(CActive::EPriorityStandard, *this);
    ReportStateAndTime();
}

void CAudioRecorder::TimerExpired(TAny* /*aTimer*/, TInt aError)
{
    ReportStateAndTime();
    if(iExampleTimer && aError != KErrCancel)
        iExampleTimer->After(KReFreshTimeOut);
}

void CAudioRecorder::RecordL(const TDesC& aFileName)
{
    iCurrentFile.iName.Copy(aFileName);

    if(iExampleTimer)
        iExampleTimer->Cancel();

    if(iToneUtility)
    {
        iToneUtility->Stop(); // stop any play/rec
        iToneUtility->Close(); // close previously opened file.
    }

    delete iToneUtility, iToneUtility = NULL;
    iToneUtility = CMdaAudioRecorderUtility::NewL(*this);

    // and free ther reserved resources.
    delete iFormat, iFormat = NULL;
    delete iCodec, iCodec = NULL;
    delete iSettings, iSettings = NULL;

    // if the file exists, we append sound data to it.
    if(BaflUtils::FileExists(CCoeEnv::Static()->FsSession(), aFileName))
        iToneUtility->OpenFileL(iCurrentFile.iName);
    else
    {
```

Record_audio_files

```
BaflUtils::EnsurePathExistsL(CCoeEnv::Static()->FsSession(),iCurrentFile.iName);

// record new Wav sound file.
iFormat = new (ELeave) TMdaWavClipFormat;
iCodec = new (ELeave) TMdaWavCodec();

iSettings = new (ELeave) TMdaAudioDataSettings;
iSettings->iSampleRate = 8000;
iSettings->iChannels = 1;// mono
iToneUtility->OpenL(&iCurrentFile,&iMdaWavClipFormat,NULL,NULL);
}

if(iExampleTimer)
    iExampleTimer->After(KRefreshTimeOut);
}

void CAudioRecorder::StopL(void)
{
    if(iExampleTimer)
        iExampleTimer->Cancel();

    if(iToneUtility)
        iToneUtility->Stop();

    ReportStateAndTime();
}

void CAudioRecorder::SetVolume(TInt& aVolume)
{
    if(aVolume < 1)
        aVolume = 1;
    else if(aVolume > 10)
        aVolume = 10;

    iVolume = aVolume;// save to internal value always
    if(iToneUtility) // and if utility exists, set it to it as well.
    {
        TInt Vol = ((iToneUtility->MaxVolume() * iVolume) / 10);
        iToneUtility->SetVolume(Vol);
    }
}

void CAudioRecorder::ReportStateAndTime(void)
{
    TInt CurrPosition(0),FileDuration(0);
    CMdaAudioClipUtility::TState CurrState(CMdaAudioClipUtility::ENotReady);

    if(iToneUtility)
    {
        CurrState = iToneUtility->State();

        TInt64 HelpPos = iToneUtility->Position().Int64();
        CurrPosition = HelpPos / 1000000;

        HelpPos = iToneUtility->RecordTimeAvailable().Int64();
        FileDuration = HelpPos / 1000000;
    }
    iObserver.StateUpdate(CurrState,CurrPosition,FileDuration);
}

void CAudioRecorder::MoscoStateChangeEvent(CBase* aObject, TInt aPreviousState, TInt aCurrentState)
```

Record_audio_files

```
{
    if(aObject == iToneUtility)
    {
        ReportStateAndTime();
        switch(aCurrentState)
        {
            case CMdaAudioClipUtility::EOpen:
            {
                if(aPreviousState == CMdaAudioClipUtility::ENotReady)
                {
                    TInt Vol = ((iVolume * iToneUtility->MaxVolume()) / 10);
                    iToneUtility->SetVolume(Vol);

                    TRAPD(err, iToneUtility->SetGain(iToneUtility->MaxGain()));
                    iToneUtility->RecordL();
                }
            } break;

            case CMdaAudioClipUtility::EPlaying:
            case CMdaAudioClipUtility::ERecording:
            case CMdaAudioClipUtility::ENotReady:
            default: // no need to do anything on these states.
                break;
        }
    }
}
```

AudioRecorder.h

```
#ifndef AUDIORECORDER_H__
#define AUDIORECORDER_H__

#include <MdaAudioSampleEditor.h>
#include <Mda\Client\Utility.h>
#include "CExampleTimer.h"

//This code has been corrected by Michel David. At least, it now compiles!!
class MExmapleRecStateObserver
{
public:
    virtual void StateUpdate(CMdaAudioClipUtility::TState aState, TInt aPosition, TInt aDurat
};

class CAudioRecorder : public CBase, public MMdaObjectStateChangeObserver, public MExampleTimerNo
{
public:
    static CAudioRecorder* NewL(MExmapleRecStateObserver& aObserver);
    static CAudioRecorder* NewLC(MExmapleRecStateObserver& aObserver);
    ~CAudioRecorder();

public: // public functions
    void RecordL(const TDesC& aFileName);
    void StopL(void);
    void SetVolume(TInt& aVolume);

private:// internal functions
    void ReportStateAndTime(void);
    void ConstructL();
    CAudioRecorder(MExmapleRecStateObserver& aObserver);
```

Record_audio_files

```
protected: // from MMdaObjectStateChangeObserver & MExampleTimerNotify
    void MoscoStateChangeEvent(CBase* aObject, TInt aPreviousState, TInt aCurrentState, TInt
    void TimerExpired(TAny* aTimer, TInt aError);

private:
    MExmapleRecStateObserver& iObserver;
    CMdaAudioRecorderUtility* iToneUtility;
    TInt iVolume;
    TMdaWavClipFormat iMdaWavClipFormat;
    TMdaFileClipLocation iCurrentFile;
    CExampleTimer* iExampleTimer;
    TMdaClipFormat* iFormat;
    TMdaPackage* iCodec;
    TMdaAudioDataSettings* iSettings;
};

#endif /*AUDIORECORDER_H__*/
```

Links

[Audio Recording APIs](#)

[Recording audio with stream](#)

[MMF](#)