



CStreamRecorder illustrates how to record audio PCM data into streams, using CMdaAudioInputStream. With this example audio is recorded as PCM data into byte buffer, if you want to use other encodings, you need to convert the PCM data into the desired format.

CMdaAudioInputStream requires the recorder class to implement MMdaAudioInputStreamCallback callback interface, from which the MaiscOpenComplete is called when the initialization for the input stream is completed, recording should never be started until this function has been called by the input stream.

As shown in the example implementation for the MaiscOpenComplete, the recording is started by calling ReadL-function of the input stream, this will record a small while which after MaiscBufferCopied is called with the recorded buffer, which in the example is then appended to the sound data and new sample recording is started by calling the ReadL-function again.

Stream_Record.cpp

```
#include <mda\common\audio.h>
#include <MdaAudioInputStream.h> // audio input stream
#include <MdaAudioOutputStream.h> // audio output stream
#include <BAUTILS.H>
#include "Stream_Record.h"

const TInt KStreamBufferSize = 320;
const TInt KStreamBufferCount = 2;

const TInt KMaxFileSize = 500000;

CStreamRecorder* CStreamRecorder::NewL()
{
    CStreamRecorder* self = CStreamRecorder::NewLC();
    CleanupStack::Pop(self);
    return self;
}

CStreamRecorder* CStreamRecorder::NewLC()
{
    CStreamRecorder* self = new (ELeave) CStreamRecorder();
    CleanupStack::PushL(self);
    self->ConstructL();
    return self;
}

CStreamRecorder::CStreamRecorder()
:iInputStream(NULL), iStreamIdx(0)
{
}

CStreamRecorder::~CStreamRecorder()
{
    if(iInputStream)
    {
        iInputStream->Stop();
        delete iInputStream;
    }
}
```

Recording_audio_with_stream

```
    }

    if(iSoundData)
    {
        delete iSoundData;
    }

    iStreamBuffer.ResetAndDestroy();
}

void CStreamRecorder::ConstructL()
{
    iReady = EFalse;
    iInputStream = CMdaAudioInputStream::NewL(*this);

    TDes8* buffer;
    for (TInt idx=0; idx<KStreamBufferCount; idx++)
    {
        buffer = new(ELeave) TBuf8<KStreamBufferSize>;
        CleanupStack::PushL(buffer);
        buffer->SetMax();
        buffer->Zero();

        User::LeaveIfError(iStreamBuffer.Append(buffer));
        CleanupStack::Pop(buffer);
    }
}

HBufC8* CStreamRecorder::GetAudioBuffer(void)
{
    HBufC8* Tmp = iSoundData;

    Stop();

    iSoundData = NULL;
    iReady = EFalse;

    return Tmp;
}

void CStreamRecorder::Record()
{
    iReady = EFalse;
    iPlayError = KErrNone;

    if(iSoundData)
    {
        delete iSoundData;
        iSoundData = NULL;
    }

    iSoundData = HBufC8::NewL(KMaxFileSize);

    if(iInputStream)
    {
        iInputStream->Stop();
        delete iInputStream;
        iInputStream = NULL;
    }

    iInputStream = CMdaAudioInputStream::NewL(*this);
```

Recording_audio_with_stream

```
iInputStream->Open(&iStreamSettings);
}

void CStreamRecorder::Stop()
{
    iReady = EFalse;

    if(iInputStream)
        iInputStream->Stop();

    // we can't delete iInputStream yet!
    // it will get cleaned up on destruction,
    // or when we start recording something new.
    //delete iInputStream;
    //iInputStream = NULL;
}

void CStreamRecorder::MaiscOpenComplete(TInt aError)
{
    iPlayError = aError;
    if (aError==KErrNone && iInputStream)
    {
        iReady = ETrue;

        iStreamSettings.iSampleRate = TMdaAudioDataSettings::ESampleRate8000Hz;
        iStreamSettings.iChannels = TMdaAudioDataSettings::EChannelsMono;

        iInputStream->SetAudioPropertiesL(iStreamSettings.iSampleRate,iStreamSettings.iChannels);
        iInputStream->SetGain(iInputStream->MaxGain());
        iInputStream->SetPriority(EPriorityNormal, EMdaPriorityPreferenceTime);
        iStreamIdx=0;
        iInputStream->ReadL(*iStreamBuffer[iStreamIdx]);
    }
}

void CStreamRecorder::MaiscBufferCopied(TInt aError, const TDesC8& aBuffer)
{
    iPlayError = aError;
    if (aError==KErrNone && iInputStream)
    {
        if (&aBuffer==iStreamBuffer[iStreamBuffer.Count()-1])
            iStreamIdx=0;
        else
            iStreamIdx++;

        iInputStream->ReadL(*iStreamBuffer[iStreamIdx]);
        if(aBuffer.Length())
        {
            TInt ll = aBuffer.Length() + iSoundData->Des().Length();

            if(ll < KMaxFileSize)
            {
                iSoundData->Des().Append(aBuffer);
            }
            else
            {
                Stop();// avoid buffer over run
            }
        }
    }
}
}
```

Recording_audio_with_stream

```
void CStreamRecorder::MaiscRecordComplete(TInt aError)
{
    iPlayError = aError;
}
```

Stream_Record.h

```
#include <mda\common\audio.h>
#include <Mda\Client\Utility.h>
#include <Mda\Common\Resource.h>
#include <MdaAudioOutputStream.h>
#include <MdaAudioInputStream.h>
#include <mmf\server\MmfCodec.h>
#include <mmf\server\mmfdatabuffer.h>

class CStreamRecorder : public CBase, public MMdaAudioInputStreamCallback
{
public:
    static CStreamRecorder* NewL();
    static CStreamRecorder* NewLC();
    ~CStreamRecorder();
public:
    void Record();
    void Stop();
    HBufC8* GetAudioBuffer(void);
    TBool StreamReady(){ return iReady;};
private:
    virtual void MaiscOpenComplete(TInt aError);
    virtual void MaiscBufferCopied(TInt aError, const TDesC8& aBuffer);
    virtual void MaiscRecordComplete(TInt aError);
private:
    void ConstructL();
    CStreamRecorder();
private:
    CMdaAudioInputStream*    iInputStream;
    RPointerArray<TDes8>     iStreamBuffer;
    TMdaAudioDataSettings    iStreamSettings;;
    TInt                     iStreamIdx, iPlayError;
    HBufC8*                  iSoundData;
    TBool                     iReady;
};
```