# Archived:Getting sample and bit rate information for audio files using Symbian C++

**Archived:** This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

## Description

Sample and bit rate information can be retrieved for some audio formats with the `DestinationSampleRateL()` and `DestinationBitRateL()` methods from `CMdaAudioRecorderUtility`. However, files encoded with codecs like MP3, AAC, or WMA require a different approach.

## Solution

The below sample code shows how to retrieve sample and bit rate values for any supported audio file. Compiling the code requires APIs from the MMF Controller Framework. These are included in the S60 3rd Edition, Feature Pack 2 SDK. They are also available for S60 3rd Edition, Maintenance Release SDK as the MMF Extensions plug-in.

**Include files:**

```
#include <mdaaudiosampleeditor.h>
#include <mmf\common\mmfcontroller.h>
```

**Libraries:**

```
mediaclientaudio.lib
mmfcontrollerframework.lib
mmfstandardcustomcommands.lib
```

The following definitions are needed:

```
class TFileParams
  {
public:
  TFileName iPath;
private:
  TInt iReserved;
  };

typedef TPckgBuf<TFileParams> TFileConfig;

/**
* CMyAudioEngine class.
* Uses CMdaAudioRecorderUtility, RMMFController and
* MMF custom commands to retrieve additional information
* about audio files.
* (Some of the implementation omitted for simplicity).
*/
class CMyAudioEngine : public CBase,
                       public MMdaObjectStateChangeObserver
  {
  ...
public:
  void GetAudioParamsL( const TDesC& aFileName );

private:

 /**
 * from MMdaObjectStateChangeObserver
 */
 virtual void MoscoStateChangeEvent(CBase* aObject, TInt aPreviousState,
                                    TInt aCurrentState, TInt aErrorCode);

private:  // Data
  TFileConfig iSourceCfg;
  TUint iSampleRate, iBitRate;
  TMMFPrioritySettings iMMFPriority;
  RMMFController iController;
  CMdaAudioRecorderUtility* iUtility;
  };
```

Implementation of `GetAudioParamsL` and `MoscoStateChangeEvent`:

```
void CMyAudioEngine::GetAudioParamsL( const TDesC& aFileName )
  {
  iSourceCfg().iPath = aFileName;
  iMMFPriority.iPriority = EMdaPriorityNormal;
  iMMFPriority.iPref = EMdaPriorityPreferenceTimeAndQuality;

  if( !iUtility )
    {
    iUtility = CMdaAudioRecorderUtility::NewL( *this );
    // Remember to delete iUtility in destructor
    }
```


(c) Copyright Nokia 2013. All rights reserved.

```
  iUtility->Close(); // Close previous audio clip
  iUtility->OpenFileL( iSourceCfg().iPath );
  }


void CMyAudioEngine::MoscoStateChangeEvent( CBase* /*aObject*/,
                                            TInt /*aPreviousState*/,
                                            TInt aCurrentState,
                                            TInt aErrorCode )
 {
 if( aErrorCode == KErrNone &&
     aCurrentState == CMdaAudioClipUtility::EOpen )
   {
   const CMMFControllerImplementationInformation* ctlInfo = 0;
   TRAPD(err, ctlInfo = &(iUtility->
     AudioPlayControllerImplementationInformationL()));

   if(err != KErrNone)
     return;

   err = iController.Open( ctlInfo->Uid(), iMMFPriority );
   if( err == KErrNone )
     {
     TMMFMessageDestination dataSource;
     err = iController.AddDataSource( KUidMmfFileSource,
                                      iSourceCfg,
                                      dataSource );
     if( err == KErrNone )
       {
       RMMFAudioControllerCustomCommands customCommands( iController );
       customCommands.GetSourceSampleRate( iSampleRate );
       customCommands.GetSourceBitRate( iBitRate );
       // See documentation on RMMFAudioControllerCustomCommands
       // for other available methods.
       }

     iController.Close();
     }
   }
 else
   {
   // TODO: Handling of other events / errors
   }
 }
```


(c) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.