Audio Playback APIs

The MMF(Multimedia Framework) provides a simple API for audio playback that is independent of the format or type of audio being played. The same API is used for playback of both local and streaming contents and is supported by the following classes:

--**CMdaAudioPlayerUtility** - The Audio Player API that supports audio playback operations and simple metadata retrieval operations.This class provides most of the audio playback operations and supports both local audio playback and streaming audio playback. Other than audio playback operations, this class also provides methods for more advanced operations such as metadata information retrieval and audio balancing.

--**MMdaAudioPlayerCallback** - The Callback API for the CMdaAudioPlayerUtility class that reports errors and notifies the application of the status of file-open operations or when play has completed.This class is a mixing class that provides feedback to applications that make use of the CMdaAudioPlayerUtility class. It is responsible for notifying the application about any error situations that occur during file opening and playback.

These classes are found in the **MediaClientAudio.lib** library.

Header files.

```
#include <MdaAudioSamplePlayer.h>
```

Libraries.

```
LIBRARY        mediaclientaudio.lib
```

Audio Playback Operations: The following operations are done using CMdaAudioPlaybackUtility class:

- Select a file to be played.
- Start audio playback.
- Change playback volume.
- Pause playback.
- Stop playback.
- Query metadata information.

Note that since all operations in the CMdaAudioPlaybackUtility class are **asynchronous operations**, it is necessary to inherit from MMdaAudioPlayerCallback so that the audio playback operations can be observed. The observer callback API consists of the following two methods:

```
        void MapcInitComplete(TInt aError, const TTImeIntervalMicroSeconds& aDuration);
```

```
        void MapcPlayComplete(TInt aError);
```

The callback methods will be invoked after playback initialization is completed and when the audio track playback is completed. In order to carry out the playback operations, it is first necessary to create an instance of the CMdaAudioPlayerUtility class using:

```
CMdaAudioPlayerUtility* iPlayer = CMdaAudioPlayerUtility::NewL(*this);
iPlayer->OpenFileL( aFilename);
iPlayer->OpenDesL( aDescriptor);
iPlayer->OpenUrlL( aUrl );
```

The above file-open operations can be done in the other way as well(like as follows):

```
CMdaAudioPlayerUtility* iPlayer =
CMdaAudioPlayerUtility::NewFilePlayerL(aFilename, *this);

CMdaAudioPlayerUtility* iPlayer =
CMdaAudioPlayerUtility::NewDesPlayerL(aDescriptor, *this);

CMdaAudioPlayerUtility* iPlayer =
CMdaAudioPlayerUtility::NewDesPlayerReadOnlyL(aDescriptor, *this);
```

After file-open operations are completed, then playback operation can be started and playback operation properties can be set.

```
iPlayer->Play();
iPlayer->SetVolume( aVolume );
iPlayer->SetRepeats( aRepeatNumberOfTimes, aTrailingSilenceMicroSeconds );
iPlayer->SetVolumeRamp( aRampDuration );
iPlayer->SetPriority( aPriority );
iPlayer->SetPosition( aPositionMicroSeconds );
```

To query the playback information of the opened audio clip:

```
TTimeIntervalMicroSeconds aDuration = iPlayer->Duration();
TTimeIntervalMicroSeconds apposition = iPlayer->Position();
TInt aVolume = iPlayer->MaxVolume();
```

Playback is paused, resumed, stopped and closed using:

```
iPlayer->Pause();
iPlayer->Play();
iPlayer->Stop();
iPlayer->Close();
```

Links

Audio Recording APIs

Full Duplex Audio

Audio Proxy Server