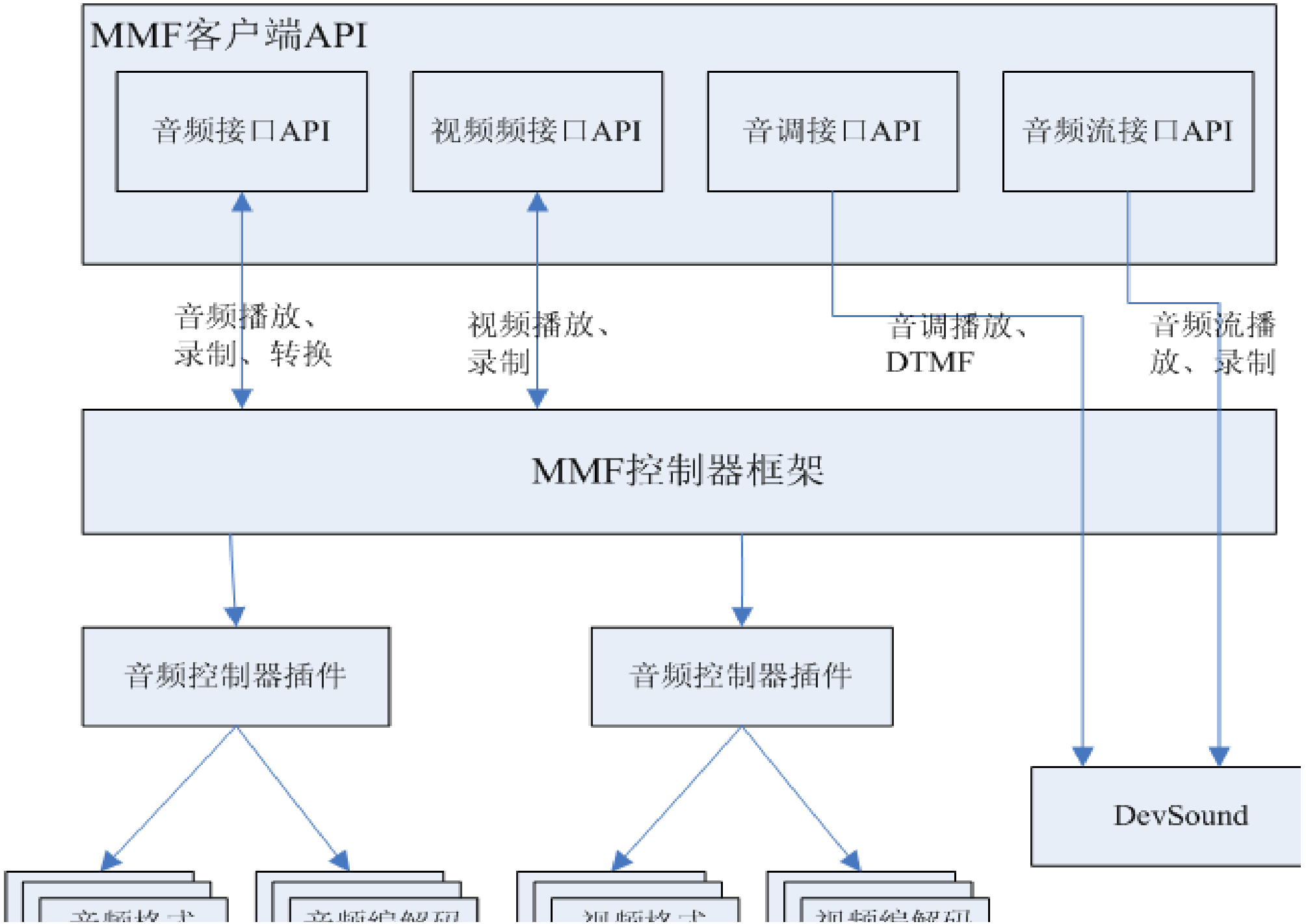
华清远见
FARSIGHT
嵌入式培训专家

嵌入式学
华清远见旗下

多媒体应用开发

架构概述(MMF客户端API)



播放音调

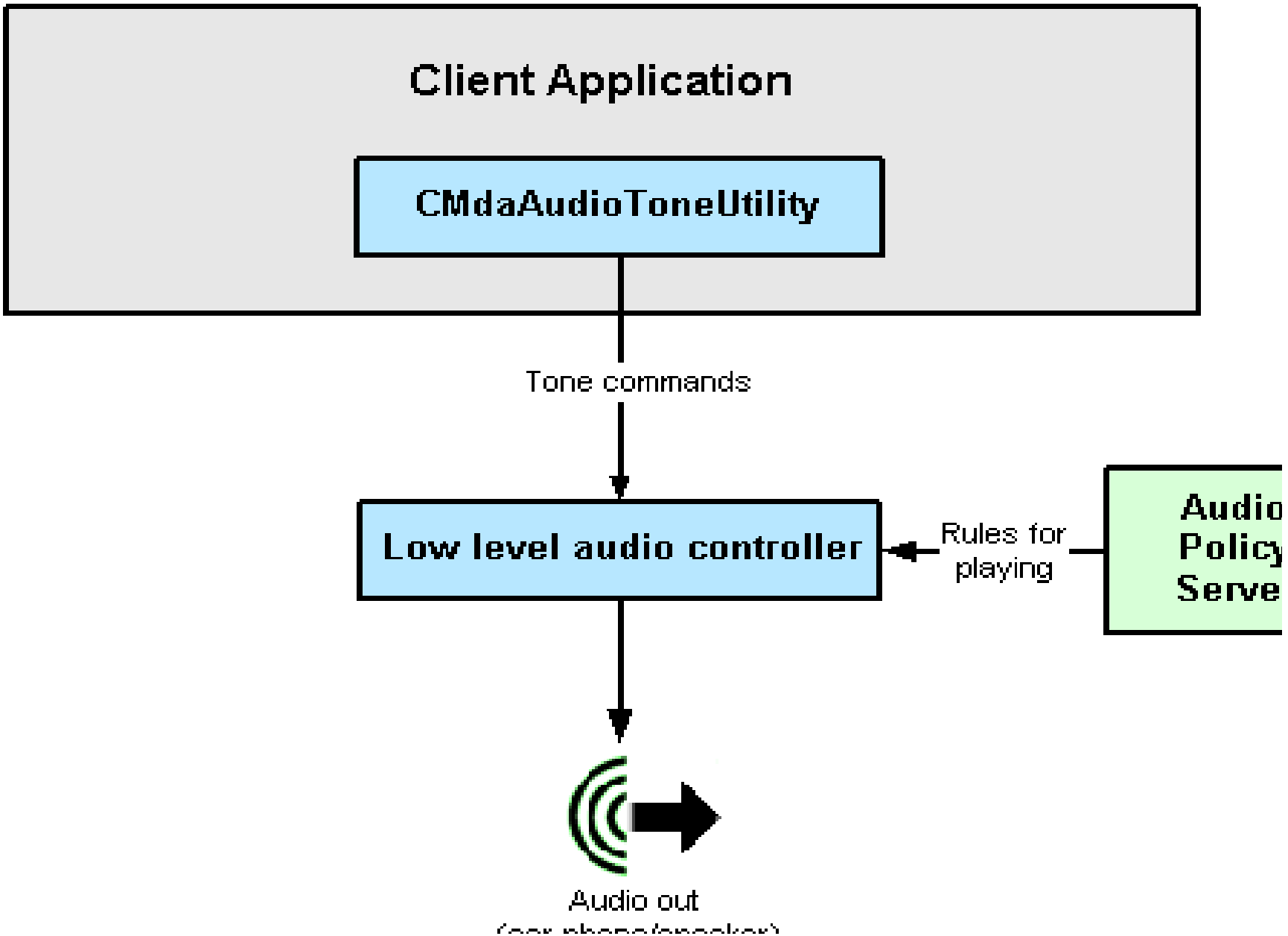
（1）指定周期和频率的简单声音。

（2）DTMF（双音多频）电话信号声音

（3）存储在文件或描述中的声音序列

（4）在手机中的预定义的声音序列

框架


Client Application
CMdaAudioToneUtility
Tone commands
Low level audio controller
Rules for playing
Audio Policy Serve
Audio out

番放音调工具类

CMdaAudioToneUtility

侦听器接口类

```
class MMdaAudioToneObserver
    {
public:
    virtual void MatoPrepareComplete(TInt aError) = 0;
    virtual void MatoPlayComplete(TInt aError) = 0;
    };
```

播放音调步骤

（1）创建音调声音播放工具类实例

```
static CMdaAudioToneUtility* NewL(MMdaAudioToneObserver& aObserver,
    CMdaServer* aServer = NULL);

static CMdaAudioToneUtility* NewL(MMdaAudioToneObserver& aObserver,
    CMdaServer* aServer,TInt aPriority,
    TMdaPriorityPreference aPref =
    EMdaPriorityPreferenceTimeAndQuality);
```

播放音调步骤

（2）准备音调声音数据

```
void PrepareToPlayTone(TInt aFrequency,
    const TTimeIntervalMicroSeconds& aDuration);

void PrepareToPlayDTMFString(const TDesC& aDTMF);
```

……

播放音调步骤

（3）配置播放工具类

```
//设置和获取音频平衡
void SetBalanceL(TInt aBalance=KMMFBalanceCenter);
TInt GetBalanceL();
//设置和获取音频设备的音量
virtual TInt Volume();
virtual void SetVolume(TInt aVolume);
virtual TInt MaxVolume();
virtual void SetVolumeRamp(const TTimeIntervalMicroSeconds& aRampDuration);
virtual void SetPriority(TInt aPriority, TMdaPriorityPreference aPref);
virtual void SetDTMFLengths(TTimeIntervalMicroSeconds32
    aToneLength,TTimeIntervalMicroSeconds32 aToneOffLength,   TTimeIntervalMicroSeconds3
    aPauseLength);
```

番放音频剪辑工具类

CMdaAudioPlayerUtility

侦听器接口类

```
MMdaAudioPlayerCallback
  {
public:
  virtual void MapcInitComplete(TInt aError,
        const TTimeIntervalMicroSeconds& aDuration) = 0;
  virtual void MapcPlayComplete(TInt aError) = 0;
  };
```

播放音频剪辑基本步骤

（1）创建音频播放工具类的实例

- static CMdaAudioPlayerUtility* NewL(MMdaAudioPlayerCallback& aCallback,TInt aPriority = aPriorityNormal,TMdaPriorityPreference aPref = EMdaPriorityPreferenceTimeAndQuality);

- static CMdaAudioPlayerUtility* NewFilePlayerL(const TDesC& aFileName,MMdaAudioPlayerCallback& aCallback,TInt aPriority = EMdaPriorityNormal,TMdaPriorityPreference aPref = EMdaPriorityPreferenceTimeAndQuality,CMdaServer* aServer = NULL);

番放音频剪辑基本步骤

（2）打开音频文件

```
void OpenFileL(const TDesC& aFileName);
void OpenFileL(const RFile& aFile);
```

在调用OpenFileL()方法打开音频文件成功后，音频文件播放工具类会回调MMdaAudioPlayerCallback::MapcInitComplete()方法，通知调用者文件打开功，工具类实例初始化完成。

番放音频剪辑基本步骤

（3）配置音频文件播放工具类实例

```
void SetVolume(TInt aVolume);
void SetVolumeRamp(const TTimeIntervalMicroSeconds& aRampDuration);
virtual TInt MaxVolume();
TInt GetVolume(TInt& aVolume);
TInt SetBalance(TInt aBalance = KMMFBalanceCenter);
TInt GetBalance(TInt& aBalance);
```

番放音频剪辑基本步骤

（4）音频文件元数据

TInt GetNumberOfMetaDataEntries(TInt& aNumEntries);

CMMFMetaDataEntry* GetMetaDataEntryL(TInt aMetaDataIndex);

播放音频剪辑基本步骤

（5）播放音频文件

- Play() – 开始从已经打开的音频数据的当前位置播放声音
- SetRepeats() – 可以调用该方法设置播放已打开音频文件的循环次数。
- Duration() – 取得以微秒表示的打开的音频文件的音频数据的长度。
- SetPlayWindow() 和 ClearPlayWindow() – 用于操作音频文件的一个剪辑，录剪辑的开始位置和结束位置，播放音频数据时，将在这个设置的剪辑窗内进行。
- GetPosition() 和 SetPosition() – 用于检索或设置剪辑的当前的播放位置。
- Pause() – 暂停播放。
- Stop() – 停止播放。
- Close() –关闭所有相关的控制器及打开的文件。

录制音频剪辑工具类

CMdaAudioRecorderUtility

侦听器接口

```
class MMdaObjectStateChangeObserver
    {
public:
    virtual void MoscoStateChangeEvent(CBase* aObject,
                    TInt aPreviousState, TInt CurrentState,
                    TInt aErrorCode)=0;
    };
```

录制音频剪辑基本步骤

（1）首先创建录音工具类的实例

```
static CMdaAudioRecorderUtility* NewL(
    MMdaObjectStateChangeObserver& aObserver, CMdaServer* aServer = NUL
    TInt aPriority = EMdaPriorityNormal,
    TMdaPriorityPreference aPref = EMdaPriorityPreferenceTimeAndQuality);
```

（2）打开保存录音的文件

```
void OpenFileL(const TDesC& aFileName);
void OpenDesL(const TDesC8& aDescriptor);
```

录制音频剪辑基本步骤

（3）配置录音工具类

- SetAudioDeviceMode()

```
enum TDeviceMode
    {
    EDefault = 0,
    ETelephonyMixed = 1,
    ETelephonyNonMixed = 2,
    ELocal = 3
    };
```

- GetVolume()、SetVolume()、MaxVolume()
- GetGain()、SetGain()、MaxGain() 、GetSupportedBitRatesL()、DestinationBitRateL()、SetDestinationBitRateL()、GetSupportedSampleRates DestinationSampleRateL()、SetDestinationSampleRateL()

录制音频剪辑基本步骤

（4）元数据控制

```
//获取到元数据的个数
TInt GetNumberOfMetaDataEntries(TInt& aNumEntries);
//取得指定的元数据
CMMFMetaDataEntry* GetMetaDataEntryL(TInt aMetaDataIndex);
//添加一个新的元数据
void AddMetaDataEntryL(CMMFMetaDataEntry& aMetaDataEntry);
//替换一个已有的元数据
TInt RemoveMetaDataEntry(TInt aMetaDataIndex);
//删除指定的元数据
void ReplaceMetaDataEntryL(TInt aMetaDataIndex,
    CMMFMetaDataEntry& aMetaDataEntry);
```

录制音频剪辑基本步骤

（5）开始录音

RecordL()

Stop()

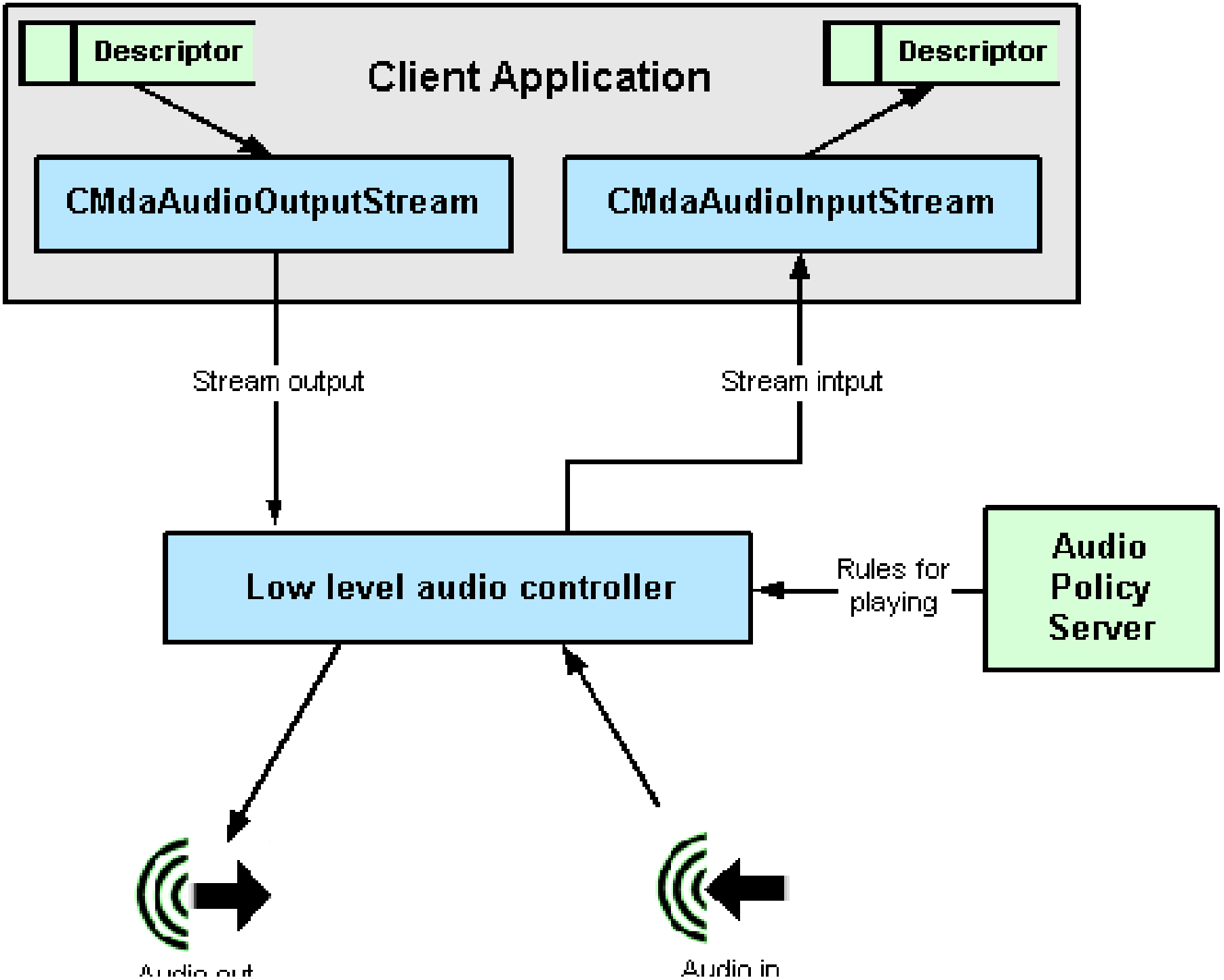
Close()

Position()

SetPosition()

CropL()

CropFromBeginningL()


Descriptor
Client Application
Descriptor
CMdaAudioOutputStream
CMdaAudioInputStream
Stream output
Stream intput
Low level audio controller
Rules for playing
Audio Policy Server
Audio out
Audio in

流式放音工具类

CMdaAudioOutputStream

侦听器接口类

```
class MMdaAudioOutputStreamCallback
    {
public:
    virtual void MaoscOpenComplete(TInt aError) = 0;
    virtual void MaoscBufferCopied(TInt aError, const TDesC8& aBuffer) = 0;
     virtual void MaoscPlayComplete(TInt aError) = 0;
    };
```

流式放音基本步骤

（1）创建音频流对象

```
static CMdaAudioOutputStream* NewL(
    MMdaAudioOutputStreamCallback& aCallBack,       CMdaServer* aServer =
    NULL);


static CMdaAudioOutputStream* NewL(
    MMdaAudioOutputStreamCallback& aCallBack,       TInt aPriority,
    TMdaPriorityPreference aPref = EMdaPriorityPreferenceTimeAndQuality);
```

流式放音基本步骤

（2）打开音频流

```
void Open(TMdaPackage* aSettings);


class TMdaAudioDataSettings : public TMdaDatatypeSettings
    {
public:
    TInt iCaps;  //指定了音频采样的能力
    TInt iMaxVolume; //音频设备的最大音量
    TInt iSampleRate; //采样速率
    TInt iChannels;          //音频采样的通道数
    TInt iVolume;            //当前音量值
    TInt iFlags;//标志，枚举型TAudioFlags值
    };
```

流式放音基本步骤

（3）设置音频流的属性

```
void SetAudioPropertiesL(TInt aSampleRate, TInt aChannels);
TInt Volume();
void SetVolume(const TInt aNewVolume);
void SetBalanceL(TInt aBalance = KMMFBalanceCenter);
TInt GetBalanceL() const;
```

流式放音基本步骤

（4）发送音频数据到MMF底层音频引擎

```
void CAudioStreamPlayer::MaoscBufferCopied(TInt aError, const TDesC8&
    /*aBuffer*/)
{
    if (aError == KErrNone)
    {
     TRAPD(err, iPlayerStream->WriteL(*iStreamBuffer))
     iPlayerStream->SetVolume(iPlayerStream->MaxVolume()/2);
    }
}
```

流式放音基本步骤

（5）停止播放

Stop() ;

流式录音工具类

CMdaAudioInputStream

侦听器接口类

```
class MMdaAudioInputStreamCallback
    {
public:
    virtual void MaiscOpenComplete(TInt aError) = 0;
    virtual void MaiscBufferCopied(TInt aError, const TDesC8& aBuffer) = 0;
    virtual void MaiscRecordComplete(TInt aError) = 0;
    };
```

视频回放工具类

CVideoPlayerUtility

侦听器接口类

```
class MVideoPlayerUtilityObserver
    {
public:
    virtual void MvpuoOpenComplete(TInt aError) = 0;
     virtual void MvpuoPrepareComplete(TInt aError) = 0;
    virtual void MvpuoFrameReady(CFbsBitmap&
      aFrame,TInt aError) = 0;
    virtual void MvpuoPlayComplete(TInt aError) = 0;
    virtual void MvpuoEvent(const TMMFEvent& aEvent)                    = 0;
    };
```

视频录制工具类

CVideoRecorderUtility

侦听器接口类

```
class MVideoRecorderUtilityObserver
    {
public:
    virtual void MvruoOpenComplete(TInt aError) = 0;
    virtual void MvruoPrepareComplete(TInt aError) = 0;
    virtual void MvruoRecordComplete(TInt aError) = 0;
    virtual void MvruoEvent(const TMMFEvent& aEvent)                    = 0;
    };
```

摄像头工具类

CCamera

侦听器接口类

```
class MCameraObserver
    {
public:
    virtual void ReserveComplete(TInt aError)=0;
    virtual void PowerOnComplete(TInt aError)=0;
    virtual void ViewFinderFrameReady(CFbsBitmap&                    aFrame)=0;
    virtual void ImageReady(CFbsBitmap* aBitmap,HBufC8* aData,TInt aError)=
    virtual void FrameBufferReady(MFrameBuffer*
      aFrameBuffer,TInt aError)=0;
    };
```

播放声音文件

谢谢！