Camera Application Engine API

Note: :This API is not part of the public SDK. It can be found in the SDK API Plug-in.

Camera Application Engine Interface API supports both image capture and video capture.An important feature available in Camera Application Engine Interface API is that it supports both still capture and burst image capture.
While making use of CCaeEngine class we will have to derive our application Camera Engine class from MCaeStillBurstObserver and MCamAppEngineObserver .These two classes provide a list of callbacks which can be used depending our use case. For example one callback method of MCamAppEngineObserver is

```
virtual void McaeoSnapImageReady(const CFbsBitmap& a Bitmap,TInt aError)
(pure virtual method)
```

which will be called asynchronously when we call CCaeEngine::CaptureStill( ) along with snap Image.Then we can make use of Image Encoder (CImageEncoder) to convert the bitmap into Jpeg and then store it in Memory.

Example code

Code for still image capture.

Single shot :
1) Derive your camera application engine class from MCamAppEngineObserver and provide implementation for all the call back methods.
2) In your application ContructL( ) have the following code snippet

```
void CCamEngine::ConstructL()
        {
 iCaeEngine=CCaeEngine::NewL();
 iCaeEngine->SetCamAppEngineObserver(*this);
 iCaeEngine->InitL();
 }
```

3) Calling CCaeEngine::InitL( ) will result in asynchronously calling MCamAppEngine::McaeoInitComplete(..) implementation.

```
void  CCamEngine::McaeoInitComplete (TInt aError)
 {
 if(!aError)
  {
  TInt err;
  TRAP(err,iCaeEngine->PrepareStillCaptureL(1));
  TRAP(err,StartViewFinderL());
  }
 else
  {
  //handle error here
  }
 }

//To Start the Camera Viewfinder
void CCamEngine::StartViewFinderL()
  {
  iCaeEngine->StartViewFinderBitmapsL(iViewSize);
  }
```


(C) Copyright Nokia 2014. All rights reserved.

Note:Calling CCaeEngine::StartViewFinderBitmapsL( ) will result in a callback to MCamAppEngineObeserver::McaeoViewFinderFrameReady(...) implementation.

```
void CCamEngine::McaeoViewFinderFrameReady (CFbsBitmap &aFrame, TInt aError)
  {
  if(!aError)
   {
   iContainer.DrawImage(aFrame);
   }
  else
   {
   //handle error
   }

  }
```

4) Then to capture a still image one should call

```
 iCaeEngine->SetSnapImageCreation(ETrue);
iCaeEngine->CaptureStill();
```

5) The call to CaptureStill will in turn asynchronously call void MCamAppEngineObserver::McaeoSnapImageReady(const CFbsBitmap& a Bitmap,TInt aError).A typical implementation would like the one given below

```
void CCamEngine::McaeoSnapImageReady (const CFbsBitmap &aBitmap, TInt aError)
 {
 SaveImage(aBitmap);
 }
void CCamEngine::SaveImage(const CFbsBitmap& aBitmap)
 {
 if(iEncoder)
  {
  delete iEncoder;
  iEncoder=NULL;
  }
 //user method to create a new filename every time we save a snap image
 GetNextUsableFileName();

 iEncoder=CImageEncoder::FileNewL(iFs,iNewFileName,KMimeType);
 iCaeEngine->StopViewFinder();
 //iSaveBmp is a CFbsBitmap object
 iSaveBmp->Duplicate(aBitmap.Handle());
 iEncoder->Convert(&iStatus,*iSaveBmp);
 SetActive();
 }
void CCamEngine::GetNextUsableFileName()
     {
        TInt index = 0;
  do{
 //iNewFileName is of type TFileName
        iNewFileName.Copy( iImagePath->Des() );
        iNewFileName.Append( KImageFileName );
        TBuf<KFileNameIndexMaxLength> num;
        num.Num( index );
```


(C) Copyright Nokia 2014. All rights reserved.

```
        iNewFileName.Append( num );
        iNewFileName.Append( KFileExtension );
        if(!BaflUtils::FileExists(CEikonEnv::Static()->FsSession(),iNewFileName))
            break;

        index ++;
        } while ( 1 );

        }
```

Burst Capture :
The difference between Still Image capture and Burst Image Capture are the following 1) We will have to call the following methods before our call to CCaeEngine::CaptureStill()

```
 iCaeEngine->SetCaeStillBurstObserver(*this);
iCaeEngine->SetStillCaptureImageCountL(numOfStillImages);
iCaeEngine->SetStillBurstCaptureIntervalL(aInterval);
```

2) We will have to implement pure virtual methods of MCaeStillBurstObserver. For more information on these classes and the method available one can refer to the documentation that comes as part of SDK Plugin Pack.

Code Snippet for Video Capture As mentioned implement all the methods of MCamAppEngineObserver. 1) Initialize the VideoRecorder

```
void CCamEngine::InitializeVideoRecordingL()
 {
 iCaeEngine->InitVideoRecorderL();
 }
```

2) After initialising the Video Recorder set the filename in which the video recording has to be saved.Then call PrepareVideoRecordingL() API.Call to PrepareVideoRecording() will in result in callback to McaeoVideoPrepareComplete() of MCamAppEngineObserver. There you can call StartRecording().Below is the code snippet of how it has been done.Similarly for stopping the videorecording you will have to call CCaeEngine::StopVideoRecording()

```
 void CCamEngine::HandleVideoRecording()
 {
 if(!iVidRecOn)//iVidRecOn is a boolean variable
  {
  iCaeEngine->SetVideoRecordingFileNameL(iNewFileName);
  iCaeEngine->PrepareVideoRecordingL(0);
  }
 else
  {
  StopRecording();
  }

 }
void CCamEngine::McaeoVideoPrepareComplete (TInt aError)
 {
 if(!aError)
  {
  StartRecording();
  }
 }
```


(C) Copyright Nokia 2014. All rights reserved.

```
void CCamEngine::StartRecording()
 {
 iCaeEngine->SetVideoAudioL(ETrue);
 iCaeEngine->StartVideoRecording();
 }

void CCamEngine::StopRecording()
 {
 iCaeEngine->StopVideoRecording();
 }
```

Example project

- CaeEngEx_V1.zip

(C) Copyright Nokia 2014. All rights reserved.