21.3.3 Video phone module

Media capture

8.CaptureDev

This class is used to find the audio and video capture devices. It has two static methods which return MediaLocators of the audio capture device and the video capture device if at least one such device is present and registered in JMF registry. It assumes that the audio capture device supports the Linear format and the video device outputs raw data in RGB format or YUV format. It queries the captureDeviceManager to find a list of devices supporting the audio or video format and returns the MediaLocator of the first encountered device in the list. Following is a sample of the text printed by CaptureDev class in the Cosmophone.log file in a typical execution of the Cosmophone.

```
Looking for audio device with the following format : LINEAR, 44100.0 Hz, 16-bit, mono
The mediaLocator for audio device  is dsound://
Looking for video device with the following format : YUV, 320x240, FrameRate=15.0
The mediaLocator for video device  is vfw://0
```

Figure 21.15 : Logging information printed by CaptureDev.class in jmf.log.

Session management

9. SessionListenerAdaptor

This class is a simple (or skeleton) sessionListener which listens to all session events and prints on System.out the details of event. This class can be adapted by extending it and overriding the appropriate event handlers.

10. session

The class session is used as a factory for creating RTPManagers. JMF sessions are managed by RTPManagers. To construct RTPManagers we need

(I) to create an instance of RTPManager,
(II) invoke the intialize() method of RTPManager by passing an array of localAddress, an array of SourceDescription, rtcpBandwidthFraction, rtcpSenderBandwidthFraction.
(III) Add SessionListener if need be.
(IV) Add target(s).

If an application has to create multiple RTPManagers the first three steps mentioned above are invariably the same in the creation of all the RTPManagers. The class session produces RTPManagers with the first three steps already performed.

The method setSourceDescription of the class session is used to give the source description details. This will be used by the class session while initializing the RTPManagers.

The videoPhone calls instantiates an object of session class, then it invokes setSourceDescription method. Then it creates an audioSessionManager and a

videoSessionManager by invoking twice the createSession and by passing IP address, port (audio port or video port) of remote machine (and ttl). The createsession method constructs a RTPManager and invokes the initSession() and the startSession().

The class session implements sessionListener interface, and in the upDate(sessionEvent) method, it delegates the update functionallity to MySessionListener which is an instance of sessionListenerAdaptor.

Note:
In later versions of the cosmophone application the session class has been removed and its functionality is included in the class videophone.

Media reception

11. ReceiveStreamListenerAdaptor

The class ReceiveStreamListenerAdaptor listens to the receiveStreams of a session and invokes the appropriate receiveStreamEvent handlers. The handlers simply print the details of the event on their occurrence. The class Receiver overrides and handles newReceiveStream events.

12. Receiver

The class Receiver handles new receive streams. Receiver extends receiveStreamListenerAdaptor and overrides the newReceiveStreamHandler method. It extracts the stream from the posted event and gets the DataSource out of the stream. Then it constructs a Processor following the different steps. First it creates a Processor and then configures it. Since the Processor has to be used as a Player to playback the received stream it sets the contentDescriptor as null.

Then by using the trackControl it checks the tracks present in the Processor (we except either an audio track or a video track). If an audio track is detected, the renderer buffer size is set before realizing and prefetching the processor. Similarly it does for the video track also.

It adds the visual and control components of the video and the control component of the audio in a single frame. The title of the frame is set with the CNAME of the sender. In the stop method the receiver stops and deallocates each of the Processor it has created.

Note:
Two receivers are created one for video and another for audio though a single receiver would suffice. However since the frame is a static variable both receiver objects adds panels to the same frame.

Media Transmission

13. sendStreamListenerAdaptor

The class sendStreamListenerAdaptor listens to the sendStream of a session and invokes the appropriate sendStreamevent handlers. The handlers simply print the details of the event on their occurrence.

14. ProcFactory

The class ProcFactory is a factory to create audioProcessor and videoProcessor for vidoPhone application. The static method createAudioProcessor returns an audioProcessor that captures audio from microphone, sets capture buffer size, and sets the audio track to GSM_RTP or G723.1_RTP format.

The static method createvideoProcessor returns a videoProcessor that captures video from web Camera, and sets the video trackformat to H263_RTP or JPEG_RTP format. The createVideoProcessor does the following:

(i) It first creates a Processor from the given DataSource. It then configures the Processor.
(ii) Then it extracts the various tracks using the track control. Then it checks for a track with videoformat. If a track with video format is found then it sets the track format to h263 or JPEG as the case may be.
(iii) The content Descriptor is then set to RAW format and the Processor is realized.

Quality control is used to set the quality factor of the video encoder. Quality is set to the value 1.0 for the H263 codec and it is set to the value 0.3 for the JPEG codec.

```
## Here's the completed flow graph:
    com.sun.media.parser.RawBufferParser@4eeaaf
      connects to: com.sun.media.codec.video.colorspace.YUVToRGB@11bfbc
      format: YUV Video Format: Size = java.awt.Dimension[width=640,height=480]
MaxDataLength = 460800 DataType = class [B yuvType = 2 StrideY = 640 StrideUV = 320
OffsetY = 0 OffsetU = 307200 OffsetV = 384000
    com.sun.media.codec.video.colorspace.YUVToRGB@11bfbc
      connects to: com.sun.media.codec.video.colorspace.RGBScaler@55c0f9
      format: RGB, 640x480, FrameRate=15.0, Length=921601, 24-bit, Masks=3:2:1,
PixelStride=3, LineStride=1920
    com.sun.media.codec.video.colorspace.RGBScaler@55c0f9
      connects to: com.sun.media.codec.video.colorspace.JavaRGBToYUV@4310d0
      format: RGB, 352x288, FrameRate=15.0, Length=304128, 24-bit, Masks=3:2:1,
PixelStride=3, LineStride=1056
    com.sun.media.codec.video.colorspace.JavaRGBToYUV@4310d0
      connects to: com.ibm.media.codec.video.h263.NativeEncoder@4fc23
      format: YUV Video Format: Size = java.awt.Dimension[width=352,height=288]
MaxDataLength = 152064 DataType = class [B yuvType = 2 StrideY = 352 StrideUV = 176
OffsetY = 0 OffsetU = 101376 OffsetV = 126720

    com.ibm.media.codec.video.h263.NativeEncoder@4fc23
      connects to: com.sun.media.multiplexer.RawBufferMux@457d21
      format: H263/RTP, 352x288, FrameRate=15.0, Length=1456
```

Figure 21.16: A typical flow graph constructed for the videoProcessor of the ProcFactory. The above was printed in the jmf.log file while executing Cosmophone.

The static method createAudioProcessor creates and returns a realized audio Processor. It is similar to createVideoProcessor. The createAudioProcessor first creates a Processor out of the audio media locator, then it sets the capture buffer size to improve the audio quality. It then

configures the Processor and extracts the tracks using trackControl. It searches for an audio track and sets the format to either GSM or G723.1 as the case may be. It then sets the ContentDescriptor as RAW. It realizes the Processor and returns it. We take care of avoiding the race condition during processor state transition.

```
## Here's the completed flow graph:
  com.sun.media.parser.RawBufferParser@3bdd48
  connects to: com.ibm.media.codec.audio.rc.RCModule@3cb23e
  format: LINEAR, 44100.0 Hz, 16-bit, Stereo, LittleEndian,Signed
    com.ibm.media.codec.audio.rc.RCModule@3cb23e
      connects to: com.ibm.media.codec.audio.g723.NativeEncoder@5a3923
   format: LINEAR, 8000.0 Hz, 16-bit, Mono, LittleEndian, Signed
    com.ibm.media.codec.audio.g723.NativeEncoder@5a3923
      connects to: com.ibm.media.codec.audio.g723.Packetizer@5d616e
      format: g723, 8000.0 Hz, Mono, FrameSize=192 bits
    com.ibm.media.codec.audio.g723.Packetizer@5d616e
      connects to: com.sun.media.multiplexer.RawBufferMux@27b0bf
    format: g723/rtp, 8000.0 Hz, 8-bit, Mono, FrameSize=192 bits
```

Figure 21.17: A typical flow graph constructed for the audioProcessor of the ProcFactory. The above was printed in the jmf.log file while executing Cosmophone.