KPR Camera

1. Introduction

This document is about how KPR camera is implemented and why it's implemented the way it is.

KPR Camera is composed of a suite of Fsk camera media readers and their common ECMAScript API. As each operating system may have its own native camera API, a Fsk camera media Reader is created for each OS as a bridge that connects the native camera API to Fsk media reader component API, which in turn interfaces with Fsk media framework. The following table shows the operating systems on which Fsk camera media readers are implemented, and the native camera APIs used.

Fsk Camera Media Reader	**Android**	**iOS**	**Mac**	**Windows**	**Linux Desktop**	**Linux Embeded**
Native Camera API	Android Java Camera API	AVFoundation	AVFoundation	DirectShow	V4L2	V4L2

2. Fsk Camera Media Reader

A camera media reader is a Fsk media reader component. As a camera typically captures a constant stream of video frames or audio samples, it works best in push mode. When media reader sends `kFskEventMediaReaderDataArrived` event to its client, this triggers an ECMAScript `onMediaDataArrived` so the application can "extract" newly arrives samples with minimum latency.

Camera functions are accessed through media reader API with several extension of camera specific media properties and data arrival callback. A camera may have a video preview stream, photo stream, video recording stream and audio stream. In a Fsk camera media reader, each of these streams is treated as a media track. And each track can have multiple resolutions or sample rates. In addition, some existing media track properties can be used to implement camera specific functions. For example, to take pictures, the photo track is "enabled" to simulate pressing the shutter.

3. Constants and Media Properties for Camera

3.1. Camera Media Mime Types

```
"video/x-kinoma-capture"
```

3.2. Camera Track Media Types

Video Preview Track: `"video-preview"`
Photo Track: `"image"`
Video Track: `"video"`
Audio Track: `"audio"`

*Video, Audio tracks are not available for this release.

3.3. New Media Properties for Camera

Media properties implemented for camera:

```
kFskMediaPropertyLens = 0xa001,    // string "back", "front"
kFskMediaPropertyAutoFocusState = 0xa002, // integer 0: not
focused, not focusing, 1: focusing, 2: focused
kFskMediaPropertyAutoFocusArea = 0xa003,  // FskRectangleRecord
kFskMediaPropertyDimensionsList = 0xa004, //integer list
```

4. ECMAScript Camera API

An ECMAScript application can directly access camera media reader through KPR ECMAScript camera API.

To create a camera:

```
var cam = new FskMediaReader( "", "video/x-kinoma-capture" );
```

To choose camera:

```
var camera_name = "back"; // or "front"
cam.set(FskMediaProperty.lens, camera_name);
```

To start camera:

```
cam.start();
```

To stop camera:

```
cam.stop();
```

To get camera video preview track:

```
var track, trackIndex = 0;
while ( track = reader.getTrack( trackIndex++ ) ) {
    var mediaType = track.get( FskMediaProperty.mediaType );
        if ( "video-preview" == mediaType )
          break;
}
if ( !track )
    throw "No video preview track";
else
     video_preview_track = track;
```

Similarly you can get video, photo and audio tracks.

To get camera video preview track dimensions lists:

```
var dim_list =
video_preview_track.get(FskMediaProperty.dimensionList);
for (var i = 0; i < dimensions_list.length; i += 2){
    var dimension = {width:dim_list[i],height: dim_list[i+1]};
     //use dimension as usual
     //etc
}
```

Similarly you can get dimensions lists of video and photo tracks.

To set camera video preview track dimensions:

```
video_preview_track.set(FskMediaProperty.dimensions,
wanted_dimensions);
```

Similarly you can set video and photo track dimensions.

To set auto focus:

```
var af_state = 1;
cam.set(FskMediaProperty.autoFocusState, af_state);
```

The auto focus area is defined by it’s coordinates x, y and dimension width, height. The coordinates range from -1000 to 1000, with (0,0) being the center, (-1000,-1000) being the top left corner and (1000, 1000) being the lower right corner of the frame. Width and height have to be greater than 0.

To get auto focus area:

```
var focus_area = cam.get(FskMediaProperty.autoFocusArea);
```

To set auto focus area, a translation between display coordinates and the camera sensor coordinates needs to be performed.

```
//Assuming the display has a dimension of 1280x720
//and the center of auto focus area is (400,450)
var display_width = 1280;
var display_height = 720;
var focus_display_x = 400;
var focus_display_y = 450;

//To calculate the coordinates on sensor
var focus_sensor_x = focus_display_x*2000/1280 - 1000; // -375
var focus_sensor_y = focus_display_y*2000/720 - 1000; // 250

//Assuming auto focus area has both width and height as 100
focus_area.width = 100;
focus_area.height = 100;

//Offset x and y by half of width and height
focus_area.x = focus_sensor_x - (focus_area.width/2);  //-425
focus_area.y = focus_sensor_y - (focus_area.height/2); //200

//Set the auto focus area
cam.set(FskMediaProperty.autoFocusArea,focus_area);
```

To capture a photo, first get photo track and then enable the track:

```
photo_track.set( FskMediaProperty.enabled, true );
```

When it’s done, disable the track:

```
photo_track.set( FskMediaProperty.enabled, false );
```

Camera media sample arriving callback:

```
<method id="onMediaDataArrived"><![CDATA[
application.distribute( "onDataArrived" );
]]></method>

<method id="onDataArrived" params="container"><![CDATA[
//extract camera sample data
var data = cam.extract();

//display a photo assuming the container is “picture”
if ( "image" == data.track.get( FskMediaProperty.mediaType ) ){
  var photo_formatInfo = data.track.get(
FskMediaProperty.formatInfo );
  var photo_format = data.track.get( FskMediaProperty.format );
  var photo_dimensions = data.track.get(
FskMediaProperty.dimensions );
  container.load( data.chunk, photo_format, photo_formatInfo );
  data.chunk.free();
}
return true;
]]></method>
```

Photo data format:

```
JPEG
```

5. Features Coverage

The following table shows the implemented camera features on the platforms that are supported as of 6/1/2014:

	Android	iOS	Mac	Window	Linux Desktop	Linux Embeded
Preview	x	x	x	x	x	x
Take Picture	x	x	x	x	x	x
Set Autofocus	x	x	n/a	n/a	x	x
Set/Get Focus Area	x	x	n/a	n/a	n/a	n/a
Set/Get Dimension	x	x	x	x	x	x
Set/Get Camera	x	x	x	x	x	x

6. Camera support in Kinoma Studio

Android

iOS

Mac OS X

Windows

Linux

*Please refer to the KPR “camera” example application for further details. The “camera” example application demonstrates how to display a photo preview and save a captured image to a JPEG file.

7. Todo

The following is a proposed new features list for next round of work:

```
kFskMediaPropertyAutoExposureState = 0xa005, //integer 0: not
metered, not metering, 1: metering, 2: metered
kFskMediaPropertyAutoExposureArea = 0xa006, //FskRectangleRecord
kFskMediaPropertyWhiteBalance = 0xa007, //integer list 0: auto, 1:
incandescent, 2: fluorescent, 3: daylight, 4: cloudy-daylight
kFskMediaPropertyWhiteBalanceList = 0xa008, //integer list
kFskMediaPropertyISO = 0xa009, // integer, 0: auto, other values:
ISO value
kFskMediaPropertyFlashMode = 0xa00a, //integer list 0: auto, 1:
off, 2: on, 3: torch
kFskMediaPropertyFlashModeList = 0xa00b, //integer list
```

Linux Embedded camera support in Kinoma Studio.

Audio capture