MOST Web Documentation

Release 1.0.1

Healthcare Flows - CRS4

May 02, 2017

Contents

The MOST project aims to achieve an open, modular and scalable solution for the creation, execution and management of remote clinical consultations with direct interaction between specialists.

The project consists of a set of frameworks that deal with different aspects and technologies useful for the creation of telemedicine applications.

CHAPTER 1

Streaming Framework

The Streaming Framework allows you to create desktop and mobile application capable of retrieve, decode and display AV stream transmitted by medical devices and network cameras.

The framework is based on opensource library GStreamer (http://gstreamer.com). GStreamer is a library for constructing graphs of media-handling components. The applications it supports range from simple Ogg/Vorbis playback, audio/video streaming to complex audio (mixing) and video (non-linear editing) processing.

Applications can take advantage of advances in codec and filter technology transparently. Developers can add new codecs and filters by writing a simple plugin with a clean, generic interface.

TREE:

Getting started

Android Requirements

- **latest stable gstreamer developer release**
  - Download and install the latest release of gstreamer development package for Android at http://gstreamer.freedesktop.org/pkg/android/

iOS Requirements

- **latest stable gstreamer developer release**
  - Download latest release of gstreamer development package for iOS at http://gstreamer.freedesktop.org/pkg/ios/ - (release 1.4.0 see http://lists.freedesktop.org/archives/gstreamer-devel/2014-July/048927.html)
  - Install iOS framework (installer put framework to ~/Library/Developer/GStreamer/iPhone.sdk)

Android Most Streaming API

Contents:

Javadoc

org.crs4.most.streaming

GStreamerBackend

class **GStreamerBackend** implements SurfaceHolder.Callback, *IStream*

Constructors

GStreamerBackend

public **GStreamerBackend**(HashMap (http://docs.oracle.com/javase/6/docs/api/java/util/HashMap.html)<String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html), String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html)> *configParams*, Handler *notificationHandler*)

Methods

commitProperties

public boolean **commitProperties**(*StreamProperties properties*)

destroy

public void **destroy**()

getErrorMsg

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getErrorMsg**()

getName

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getName**()

getProperty

public Object (http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html) **getProperty**(*StreamProperty property*)

getState

public *StreamState* **getState**()

getSurfaceView

public SurfaceView **getSurfaceView**()

Returns the rendering Surface

getVideoSize

public *Size* **getVideoSize**()

loadStillImage

public boolean **loadStillImage**(String *uri*) (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html)

pause

public void **pause**()
pause the stream

play

public void **play**()
Play the stream

prepare

public void **prepare**(SurfaceView *surface*)

surfaceChanged

public void **surfaceChanged**(SurfaceHolder *holder*, int *format*, int *width*, int *height*)

surfaceCreated

public void **surfaceCreated**(SurfaceHolder *holder*)

surfaceDestroyed

public void **surfaceDestroyed**(SurfaceHolder *holder*)

surfaceFinalize

public void **surfaceFinalize**()

surfaceInit

public void **surfaceInit**(Surface *surface*)

IStream

public interface **IStream**

An IStream object represents a single audio/video stream object. You can obtain a new IStream object by calling the method *StreamingLib.createStream(java.util.HashMap,android.os.Handler)*.

Methods

commitProperties

public boolean **commitProperties**(*StreamProperties properties*)

Commit the stream properties values specified as argument

Parameters

- **properties** – the stream properties to update

Returns true if no error occurred during the update request; False otherwise

destroy

public void **destroy**()

Destroy this stream

getErrorMsg

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getErrorMsg**()

Get detailed informations about a stream error (return an empty stream if the stream is not in Stream.ERROR state)

Returns infomrations about the type of stream error

getName

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getName**()

Returns the name of this stream

getProperty

public Object (http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html) **getProperty** (*StreamProperty property*)

Reads the current value of the specified stream property

Parameters

- **property** –

Returns the value of the property

getState

public *StreamState* **getState** ()

Returns the current state of this stream

getVideoSize

public *Size* **getVideoSize** ()

Returns the current size of the video stream

loadStillImage

public boolean **loadStillImage** (String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *uri*)

Load a still image from the remote camera, provided the uri

Parameters

- **uri** – the uri pointing to the image to load

Returns `true` if no error occurred during the operation, `false` otherwise

pause

public void **pause** ()

pause the stream

play

public void **play** ()

Play the stream

prepare

public void **prepare** (SurfaceView *surface*)

Prepare the stream by providing a video surface

Parameters

- **surfaceView** – the Surface where to render the stream

StreamProperties

public class **StreamProperties**

This class collects a set of stream properties a user intend to apply to a stream.

See also: *IStream.commitProperties(StreamProperties)*

Methods

add

public *StreamProperties* **add**(*StreamProperty streamProperty*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *value*)

Add a new property

Parameters

- **streamProperty** – the stream property to update
- **value** – the value to be set for this property

Returns this StreamProperties so you can chain more properties to add

get

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **get**(*StreamProperty property*)

Get the specified property value

Parameters

- **property** – the stream property key

Returns the specified property value(or null if the key was not found)

getAll

public Properties (http://docs.oracle.com/javase/6/docs/api/java/util/Properties.html) **getAll**()

Get all added properties

Returns the added properties

remove

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **remove**(*StreamProperty streamProperty*)

Remove the specified property

Parameters

- **streamProperty** – the property to remove

Returns value of the removed property, or null if this property was not found

StreamingEventBundle

public class **StreamingEventBundle**
This class represents a container of all the informations of the events triggered by the Streaming Library.

Constructors

StreamingEventBundle

public **StreamingEventBundle**(*StreamingEventType* *eventType*, *StreamingEvent* *event*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *info*, Object (http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html) *data*)
This object contains all the informations of any event triggered by the Streaming Library.

Parameters

- **eventType** – the type of this event
- **event** – the event
- **info** – a textual information describing this event
- **data** – a generic object containing event-specific informations (the object type depends on the type of the event). In particular:

•events of type *StreamingEventType.STREAM_EVENT* contain the *IStream* object that triggered this event

Methods

getData

public Object (http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html) **getData**()
Get a generic object containing event-specific informations (the object type depends on the type of the event). Note that events of type *StreamingEventType.STREAM_EVENT* contain the *IStream* object that triggered this event

Returns a generic object containing event-specific informations

getEvent

public *StreamingEvent* **getEvent**()
Get the triggered event

Returns the event triggered by the library

getEventType

public *StreamingEventType* **getEventType**()
Get the event type

Returns the event type

getInfo

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getInfo** ()
Get a textual description of this event

Returns a textual description of this event

StreamingLib

public interface **StreamingLib**

Methods

createStream

public *IStream* **createStream** (HashMap (http://docs.oracle.com/javase/6/docs/api/java/util/HashMap.html)<String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html), String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html)> *configParams*, Handler *notificationHandler*)
This factory method provides a new IStrean instance

Parameters

- **configParams** – All needed configuration string parameters. All the supported parameters are the following:

•name: (mandatory) the name of the stream (it must be unique for stream)

•uri: (mandatory) the uri of the stream (it can be also changed later)

•latency: (optional) the preferred latency of the stream in ms (default value: 200 ms)

Parameters

- **notificationHandler** – the handler where to receive all notifications from the Library

Throws

- **Exception** (http://docs.oracle.com/javase/6/docs/api/java/lang/Exception.html) – if an error occurred during the stream initialization

initLib

public void **initLib** (Context *context*)
Initialize the streaming library. Note that you must call this method before using any other method of the library

Parameters

- **context** – the application context

Throws

- **Exception** (http://docs.oracle.com/javase/6/docs/api/java/lang/Exception.html) – if an error occurred during the library initialization

StreamingLibBackend

public class **StreamingLibBackend** implements *StreamingLib*

This class implements the *StreamingLib* interface. It internally uses the GStreamer library as backend. So, you can get a *StreamingLib* instance in the following way:

```
StreamingLib myStreamingLib = new StreamingLibBackend();
myStreamingLib.initLib(getApplicationContext());
```

Remember that, before using the library, you must call the method *initLib(Context)* to initialize it. To get a *IStream* instance you can call the *createStream(HashMap,Handler)* method:

```
HashMap stream1_params = new HashMap();
stream1_params.put("name", "Stream_1");
stream1_params.put("uri", "http://docs.gstreamer.com/media/sintel_trailer-368p.ogv
↪");
Handler notificationHandler = new Handler(this);
IStream myStream = myStreamingLib.createStream(stream_1_params,␣
↪notificationHandler);
```

See also: *StreamingLib*

Methods

createStream

public *IStream* **createStream**(HashMap (http://docs.oracle.com/javase/6/docs/api/java/util/HashMap.html)<String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html), String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html)> *configParams*, Handler *notificationHandler*)

initLib

public void **initLib**(Context *context*)

org.crs4.most.streaming.enums

PTZ_Direction

public enum **PTZ_Direction**

Enum Constants

DOWN

public static final *PTZ_Direction* **DOWN**

DOWN_LEFT

public static final *PTZ_Direction* **DOWN_LEFT**

DOWN_RIGHT

public static final *PTZ_Direction* **DOWN_RIGHT**

LEFT

public static final *PTZ_Direction* **LEFT**

RIGHT

public static final *PTZ_Direction* **RIGHT**

STOP

public static final *PTZ_Direction* **STOP**

UP

public static final *PTZ_Direction* **UP**

UP_LEFT

public static final *PTZ_Direction* **UP_LEFT**

UP_RIGHT

public static final *PTZ_Direction* **UP_RIGHT**

PTZ_Zoom

public enum **PTZ_Zoom**

Enum Constants

IN

public static final *PTZ_Zoom* **IN**

OUT

public static final *PTZ_Zoom* **OUT**

STOP

public static final *PTZ_Zoom* **STOP**

StreamMode

public enum **StreamMode**

Enum Constants

MOTION_JPEG

public static final *StreamMode* **MOTION_JPEG**

MPEG_4

public static final *StreamMode* **MPEG_4**

STILL_IMAGE

public static final *StreamMode* **STILL_IMAGE**

StreamProperty

public enum **StreamProperty**

Enum Constants

LATENCY

public static final *StreamProperty* **LATENCY**
the preferred stream latency property

NAME

public static final *StreamProperty* **NAME**
the stream name property

STATE

public static final *StreamProperty* **STATE**
the current state (read only)

URI

public static final *StreamProperty* **URI**
the stream uri property

VIDEO_SIZE

public static final *StreamProperty* **VIDEO_SIZE**
the video size

StreamState

public enum **StreamState**

Enum Constants

DEINITIALIZED

public static final *StreamState* **DEINITIALIZED**
the stream has not been initialized yet , the initialization failed or it has been deinitialized

DEINITIALIZING

public static final *StreamState* **DEINITIALIZING**
the stream has being deinitialized

ERROR

public static final *StreamState* **ERROR**
The stream is in an inconsistent state

INITIALIZED

public static final *StreamState* **INITIALIZED**
the stream was successfully initialized and it is ready to play

INITIALIZING

public static final *StreamState* **INITIALIZING**
the stream has being initialized

PAUSED

public static final *StreamState* **PAUSED**
the stream is in pause state

PLAYING

public static final *StreamState* **PLAYING**
the stream is playing

PLAYING_REQUEST

public static final *StreamState* **PLAYING_REQUEST**
a play request is sent to the stream that is preparing to start playing

StreamingEvent

public enum **StreamingEvent**
Contains all events triggered by the Streaming library

Enum Constants

STREAM_ERROR

public static final *StreamingEvent* **STREAM_ERROR**
a streaming error occurred

STREAM_STATE_CHANGED

public static final *StreamingEvent* **STREAM_STATE_CHANGED**
a stream internal state change occurred

VIDEO_SIZE_CHANGED

public static final *StreamingEvent* **VIDEO_SIZE_CHANGED**
the size of the video stream changed

StreamingEventType

public enum **StreamingEventType**

Enum Constants

LIB_EVENT

public static final *StreamingEventType* **LIB_EVENT**
generic library events

STREAM_EVENT

public static final *StreamingEventType* **STREAM_EVENT**
stream events as state changes (play, pause...) or other stream related events

org.crs4.most.streaming.ptz

PTZ_Manager

public class **PTZ_Manager**

Constructors

PTZ_Manager

public **PTZ_Manager**(Context *ctx*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *uri*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *username*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *password*)
Handles ptz commands of a remote Axis webcam. (tested on Axis 214 PTZ model)

Parameters

- **ctx** – The activity context
- **uri** – The ptz uri of the webcam
- **username** – the username used for ptz authentication
- **password** – the username used for ptz authentication

Methods

getPassword

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getPassword**()
Get the password used for the authentication

Returns the password

getUri

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getUri**()
Get the uri connection string

Returns the uri used for connecting to the webcam

getUsername

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **getUsername**()
Get the username used for the authentication

Returns the username

goTo

public void **goTo**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *preset*)
Move the webcam to the position (and/or zoom value) specified by the preset passed as argument

Parameters

- **preset** – the preset name

startMove

public void **startMove**(*PTZ_Direction direction*)
Start moving the webcam to the specified direction

Parameters

- **direction** – the direction (*PTZ_Direction.STOP*} stops the webcam)

startMove

public void **startMove**(*PTZ_Direction direction*, int *speed*)
Start moving the webcam to the specified direction and speed

Parameters

- **direction** – the moving direction
- **speed** – the speed

startZoom

public void **startZoom**(*PTZ_Zoom zoomDirection*)
Start zooming to the specified direction

Parameters

- **zoomDirection** – the zoom direction

startZoom

public void **startZoom**(*PTZ_Zoom zoomDirection*, int *speed*)
Start zooming to the specified direction and speed

Parameters

- **zoomDirection** – the zoom directiom
- **speed** – the zoom speed

stopMove

public void **stopMove**()
Stop the pan and/or tilt movement of the webcam

stopZoom

public void **stopZoom**()
Stop the zoom

zoom

public void **zoom**(int *value*)
Zoom the webcam to the specified value (positive values are for zoom-in, negative values for zoom-out)

Parameters

- **value** –

org.crs4.most.streaming.utils

ImageDownloader

public class **ImageDownloader**
This class provides utility methods for downloading, storing and loading images from/to the internal storage. Note that the image downloading process is asynchronous, so it uses the interface *ImageDownloader.IBitmapReceiver* for notifying the user about the downloading outcomes.

Fields

receiver

IBitmapReceiver **receiver**

Constructors

ImageDownloader

public **ImageDownloader**(*IBitmapReceiver receiver*, Context *ctx*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *username*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *password*)
This class handles asynchronous image downloads. Once a image has been successfully downloaded, it calls the method `ImageDownloader.IBitmapReceiver.onBitmapReady(Bitmap)`}

Parameters

- **receiver** – the object that will receive the downloaded image
- **ctx** – the activity context
- **username** – the user name (needed for authentication)
- **password** – the password (needed for authentication)

Methods

deleteInternalFile

public static boolean **deleteInternalFile**(Context *ctx*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *filename*)

Deletes an image from the internal storage

Parameters

- **ctx** – the context the activity context
- **filename** – the name (no path))of the file to be deleted

Returns true if the file was deleted , false otherwise

downloadImage

public void **downloadImage**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *url*)

asynchronously downloads an image from the web

Parameters

- **url** – the url image

getInternalImages

public static File (http://docs.oracle.com/javase/6/docs/api/java/io/File.html)[] **getInternalImages**(Context *ctx*)

Get the list of .jpeg images stored in the internal archive (private to the activity context passed as argument)

Parameters

- **ctx** – the activity context

Returns the list of .jpg images

getInternalImages

public static File (http://docs.oracle.com/javase/6/docs/api/java/io/File.html)[] **getInternalImages**(Context *ctx*, String (http://docs.oracle.com/javase/ *filter*)

Get the list of images stored in the internal archive (private to the activity context passed as argument)

Parameters

- **ctx** – the activity context
- **filter** – the image type filter (e.g: ".jpg")

Returns the list of images

loadImageFromInternalStorage

public void **loadImageFromInternalStorage**(String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *filename*)

Loads an image from the internal storage. The loaded bitmap will be sent to the callback method *IBitmapReceiver.onBitmapDownloaded(ImageDownloader,Bitmap)*

Parameters

- **filename** – the name of the file to load

logAppFileNames

public void **logAppFileNames**()

saveImageToInternalStorage

public void **saveImageToInternalStorage**(Bitmap *image*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.htm *filename*)

Save a bitmap into to internal storage

Parameters

- **image** – the bitmap to save
- **filename** – the filename of the saved bitmap

ImageDownloader.IBitmapReceiver

public interface **IBitmapReceiver**

An ImageDownloader user must provide this interface for receiving notifications about image donwloading and storing.

Methods

onBitmapDownloaded

public void **onBitmapDownloaded**(*ImageDownloader imageDownloader*, Bitmap *image*)

Called when a bitmap was successfully downloaded

Parameters

- **imageDownloader** – the ImageDownloader that has triggered this callback
- **image** – the downloaded bitmap

onBitmapDownloadingError

public void **onBitmapDownloadingError**(*ImageDownloader imageDownloader*, Exception (http://docs.oracle.com/javase/6/docs/api/java/lang/Exception.html) *ex*)

Called when the image downloading failed for some raison

Parameters

- **imageDownloader** – the ImageDownloader that has triggered this callback
- **ex** – the exception raised during the downloading process

onBitmapSaved

public void **onBitmapSaved**(*ImageDownloader imageDownloader*, String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) *filename*)

Called when the bitmp was stored in to the internal storage

Parameters

- **imageDownloader** – the ImageDownloader that has triggered this callback
- **filename** – the name of the stored file

onBitmapSavingError

public void **onBitmapSavingError**(*ImageDownloader imageDownloader*, Exception (http://docs.oracle.com/javase/6/docs/api/java/lang/Exception.html) *ex*)

Called when the image saving failed for some raison

Parameters

- **imageDownloader** – the ImageDownloader that has triggered this callback
- **ex** – the exception raised during the saving process

Size

public class **Size**

Image size (width and height dimensions).

Constructors

Size

public **Size**(int *w*, int *h*)

Sets the dimensions for pictures.

Parameters

- **w** – the photo width (pixels)
- **h** – the photo height (pixels)

Methods

equals

public boolean **`equals`** (Object (http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html) *obj*)
Compares `obj` to this size.

Parameters

- **`obj`** – the object to compare this size with.

Returns `true` if the width and height of `obj` is the same as those of this size. `false` otherwise.

getHeight

public int **`getHeight`** ()

getWidth

public int **`getWidth`** ()

hashCode

public int **`hashCode`** ()

toString

public String (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) **`toString`** ()

Examples

Android

- **DualStreamingExample** located into the folder *examples/android*. This example shows how to simultaneously play two streams on two video surfaces inside an Android Activity by using the underlying native gstreamer backend.

  For running the Android example, import the project in your preferred IDE (e.g Eclipse) and do the following chang

  - Edit the file *jni/Android.mk* and properly change the absolute path of the environment variables GSTREAMER_SDK_ROOT_ANDROID and GSTREAMER_ROOT
  - Create your *uri.properties.default* property file and put it into the *assets* folder.(That folder already contains the *uri.properties* file that you can use as template for your own property file)
  - Build the project (Note that the NDK must be installed and configurated on your system in order to build the project)
  - Deploy the application on your android device or on your emulator

Other examples are included into the *android/examples* folder. Finally, consider using the MOST Visualizazion Framework (https://github.com/crs4/most-visualization) that allows you to include into mobile applications visual and interactive widgets capable of rendering and inspecting A/V streams.

iOS

- **TestDualStream** located into the folder *examples/ios8*. This example shows how to simultaneously play two streams on two video surfaces inside an iOS Application by using the underlying native gstreamer backend.

License

```
/*!
 * Project MOST - Moving Outcomes to Standard Telemedicine Practice
 * http://most.crs4.it/
 *
 * Copyright 2014, CRS4 srl. (http://www.crs4.it/)
 * Dual licensed under the MIT or GPL Version 2 licenses.
 * See license-GPLv2.txt or license-MIT.txt
 */
```

GPL2: https://www.gnu.org/licenses/gpl-2.0.txt

MIT: http://opensource.org/licenses/MIT

Authors

Code author: Francesco Cabras <francesco.cabras@crs4.it>

Code author: Stefano Leone Monni <stefano.monni@crs4.it>

Indices and tables

- genindex
- modindex
- search

Index

P

R

S

T

U

V

Z