Tizen Native Application

Minsoo Ryu

Real-Time Computing and Communications Lab.
Hanyang University

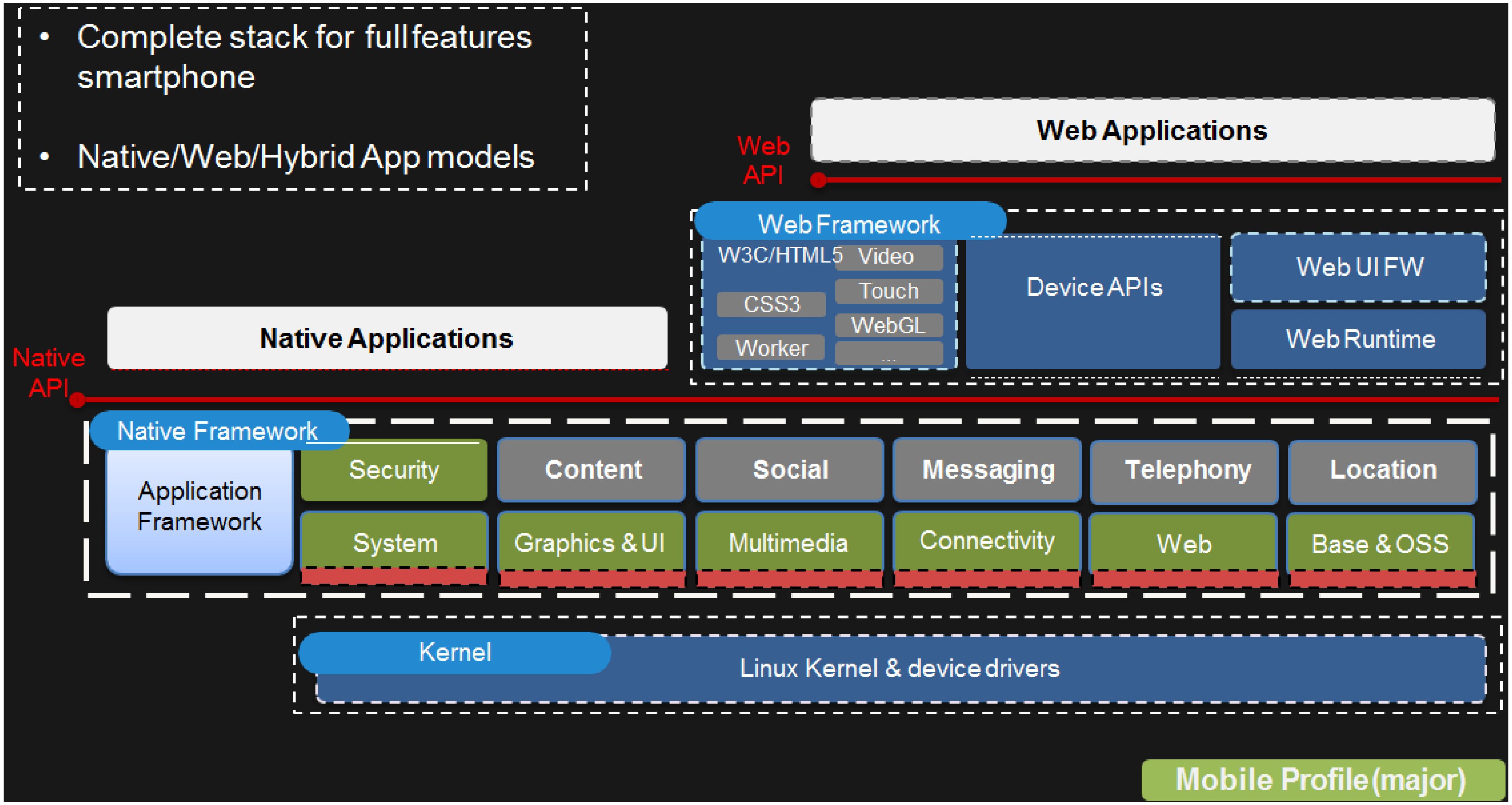
msryu@rtcc.hanyang.ac.kr

Outline

- ❒ Tizen Architecture & Application Type
- ❒ Tizen Native Application
  - Tizen Native Application Model
  - Understanding Tizen Native Application
  - Tizen Native API
- ❒ Example of Tizen Native Application - Camera
  - LAB : Tizen Native Sample Application - Camera

Tizen Architecture & Application

Tizen Architecture


- Complete stack for full features smartphone
- Native/Web/Hybrid App models

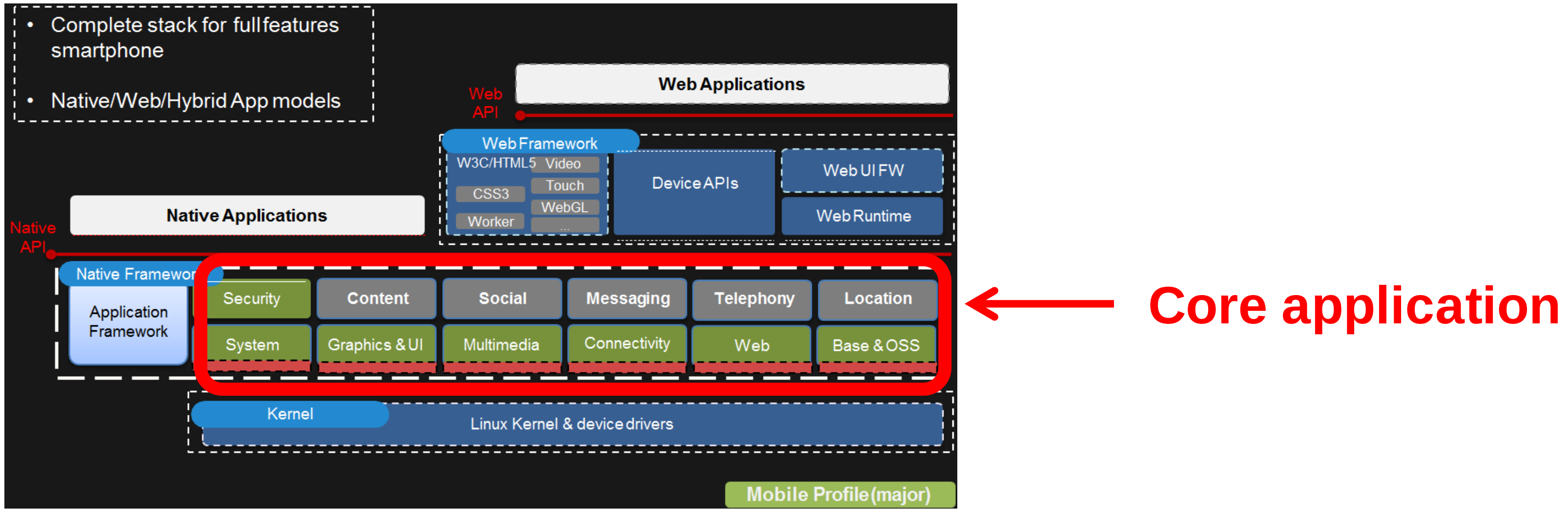
Tizen Architecture

- Tizen Public layers
  - Application development productivity
    - State-of-the-art HTML5/W3C APIs & Web UI framework
    - Full-featured native application development and features
  - Well-documented API references, developer guide, sample codes, and associated tools

- Core sub-system
  - Providing common functionalities for Web and Native framework as an underlying layer
  - Performance optimization

Tizen Application Type

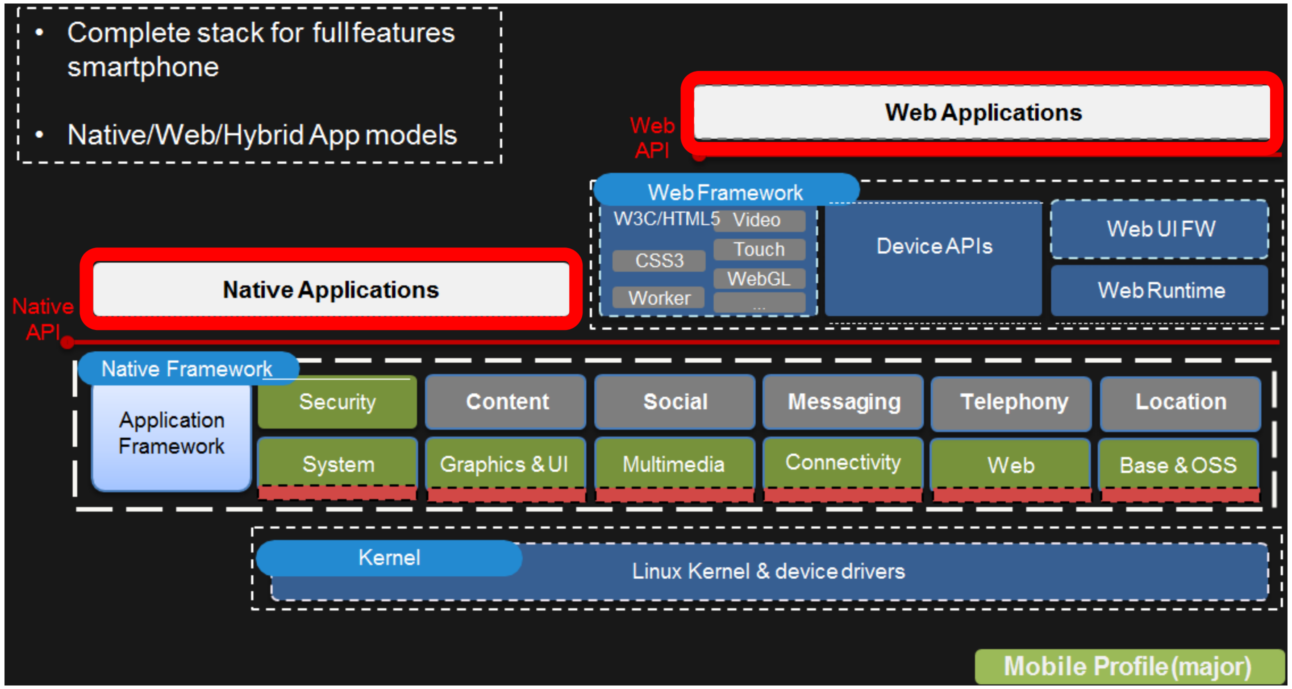
- ❒ Core Applications
  - ▪ Application using internal APIs to fully utilize device capabilities
- ❒ In-house Applications
  - ▪ Pre-loaded Core applications developed by device implementers
  - ▪ Call app, Calculator app, Gallery app, Contacts app, etc



Tizen Application Type

❒ Web & Native Application

- Apps using public API to get full support for package installation and upgrade, security, backward compatibility, and so on
- Many native sample applications included in the Tizen SDK



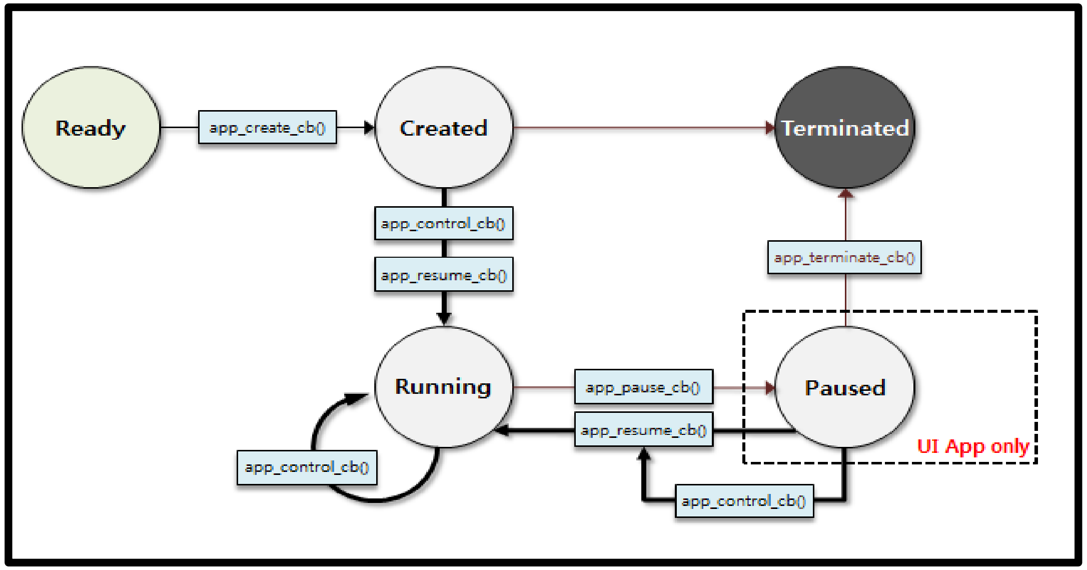
Tizen Native Application Model

Tizen Native Application Model

❐ Tizen Native Application Life Cycle

- Application States

State	Description
READY	The application is launched
CREATED	The application starts the main loop
PAUSED	The application is running but invisible to users
RUNNING	The application is running and visible to users
TERMINATED	The application is terminated

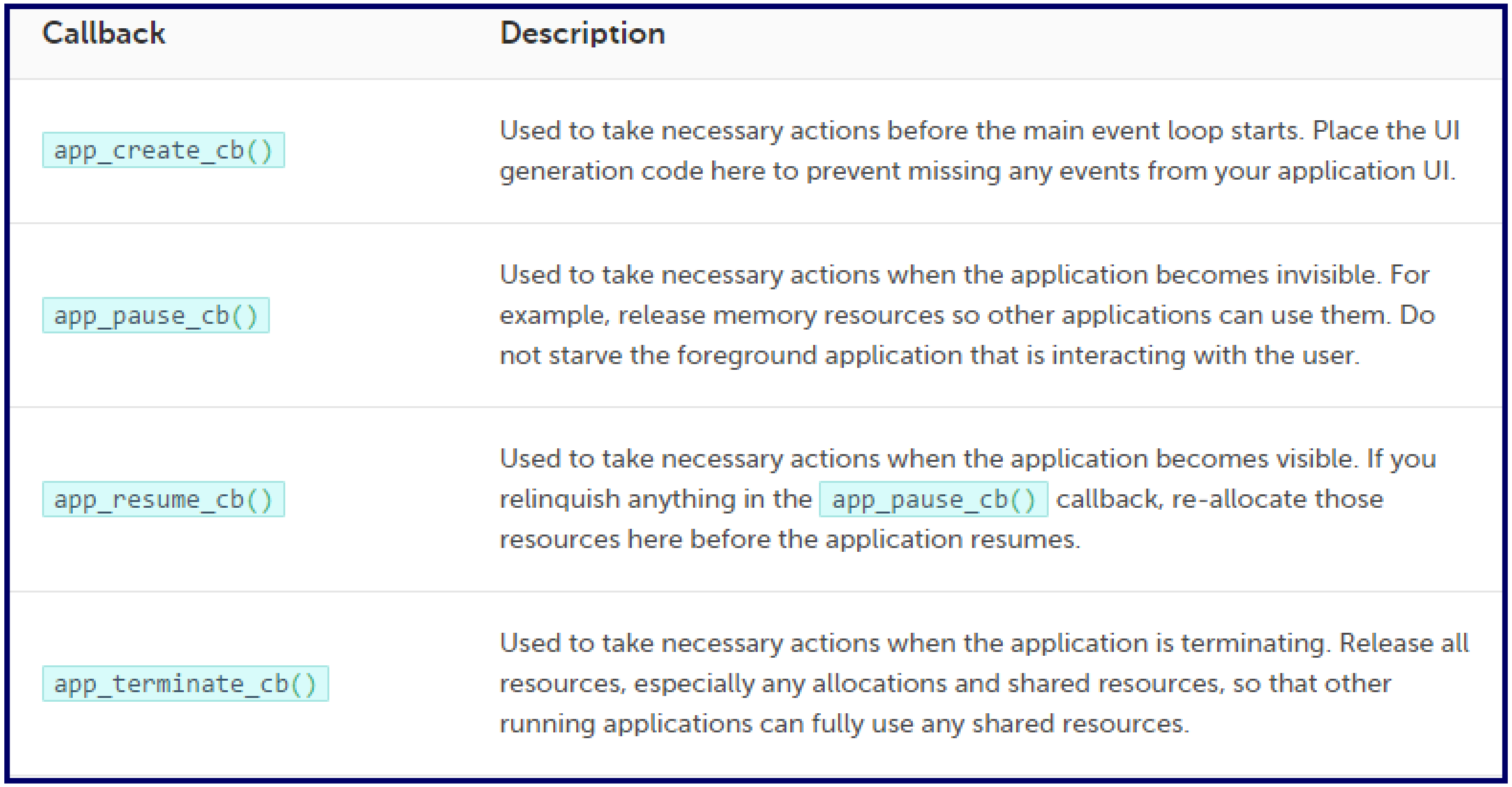


❐ State transition callbacks should be provided before starting the application main loop

Tizen Native Application Model

❒ Tizen Native Application Life Cycle

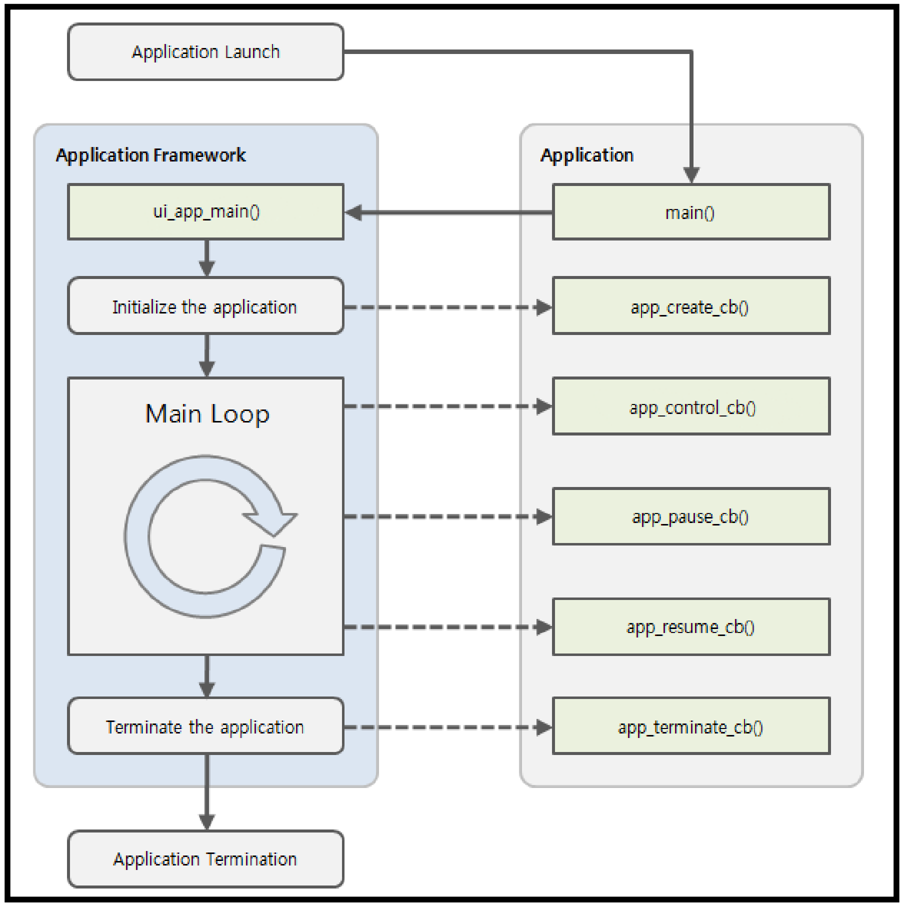
- Callbacks regarding Application Life Cycle



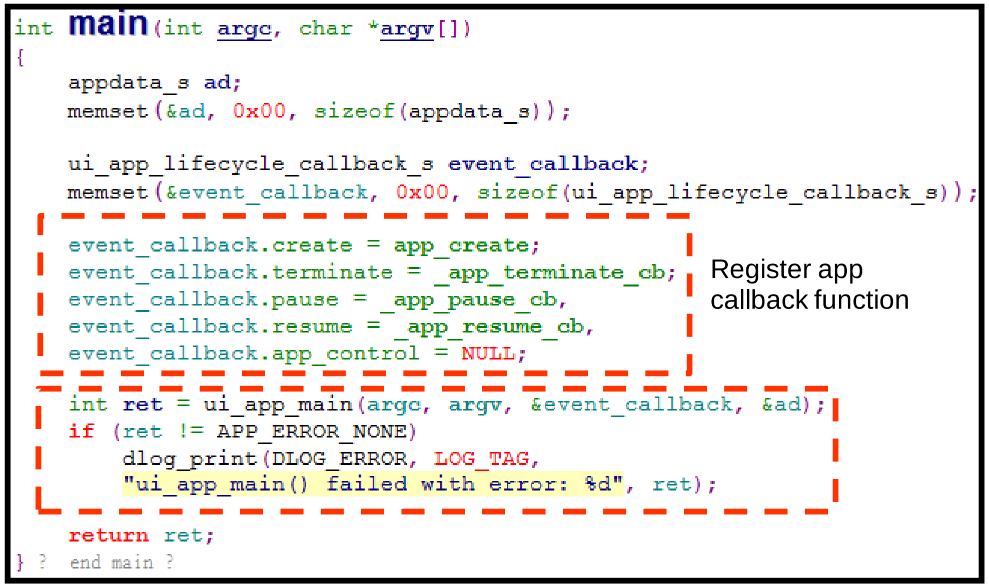
Callback	Description
`app_create_cb()`	Used to take necessary actions before the main event loop starts. Place the UI generation code here to prevent missing any events from your application UI.
`app_pause_cb()`	Used to take necessary actions when the application becomes invisible. For example, release memory resources so other applications can use them. Do not starve the foreground application that is interacting with the user.
`app_resume_cb()`	Used to take necessary actions when the application becomes visible. If you relinquish anything in the `app_pause_cb()` callback, re-allocate those resources here before the application resumes.
`app_terminate_cb()`	Used to take necessary actions when the application is terminating. Release all resources, especially any allocations and shared resources, so that other running applications can fully use any shared resources.

Tizen Native Application Model

❒ Tizen Native Application Life Cycle



- Main Function



```
int main(int argc, char *argv[])
{
    appdata_s ad;
    memset(&ad, 0x00, sizeof(appdata_s));

    ui_app_lifecycle_callback_s event_callback;
    memset(&event_callback, 0x00, sizeof(ui_app_lifecycle_callback_s));

    event_callback.create = app_create;
    event_callback.terminate = _app_terminate_cb;
    event_callback.pause = _app_pause_cb,
    event_callback.resume = _app_resume_cb,
    event_callback.app_control = NULL;

    int ret = ui_app_main(argc, argv, &event_callback, &ad);
    if (ret != APP_ERROR_NONE)
        dlog_print(DLOG_ERROR, LOG_TAG,
        "ui_app_main() failed with error: %d", ret);

    return ret;
} ? end main ?
```

Register app callback function

Understanding Tizen Native Application

Understanding Tizen Native Application

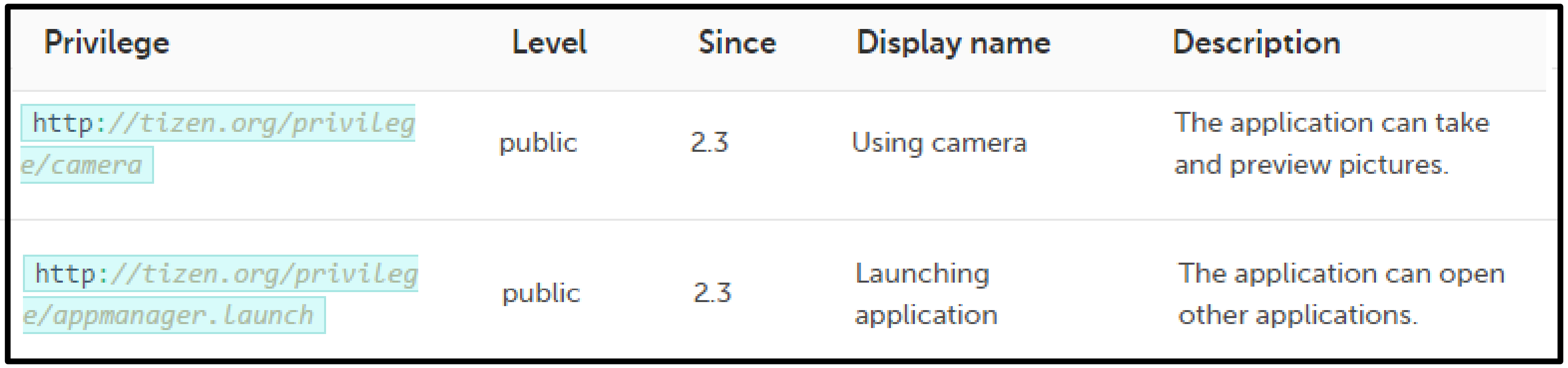
❒ Security and API Privileges

- To effectively protect the device system and user private data, the Tizen security architecture is based on privileges and application signing of the Linux basic security model, which includes process isolation and mandatory access control
- Tizen provides API-level access control for security-sensitive operations
- Applications that use such sensitive APIs *must declare the required privileges in the tizen-manifest.xml file*
- If an application invokes a privileged API, the Tizen system checks whether the privilege is present in the *tizen-manifest.xml* file
- If the privilege is not present in the file, the system prohibits the application execution

Understanding Tizen Native Application

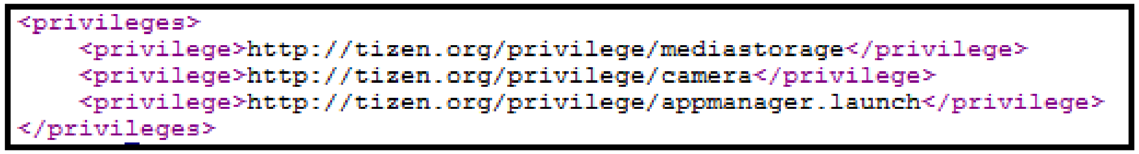
❐ Security and API Privileges

- Add Privilege for Camera using and App launching at a specific application



Privilege	Level	Since	Display name	Description
http://tizen.org/privilege/camera	public	2.3	Using camera	The application can take and preview pictures.
http://tizen.org/privilege/appmanager.launch	public	2.3	Launching application	The application can open other applications.

- Native Application manifest.xml



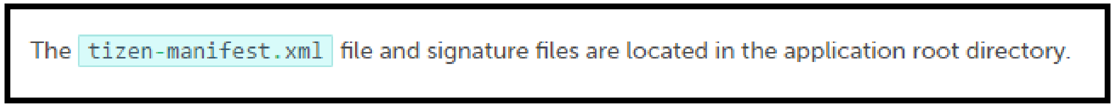
```
<privileges>
    <privilege>http://tizen.org/privilege/mediastorage</privilege>
    <privilege>http://tizen.org/privilege/camera</privilege>
    <privilege>http://tizen.org/privilege/appmanager.launch</privilege>
</privileges>
```

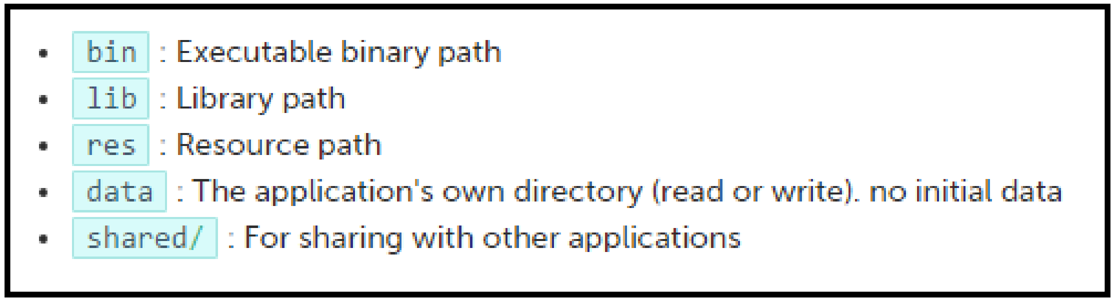
※Privileges List
https://developer.tizen.org/ko/development/getting-started/native-application/understanding-tizen-programming/security-and-api-privileges

Understanding Tizen Native Application

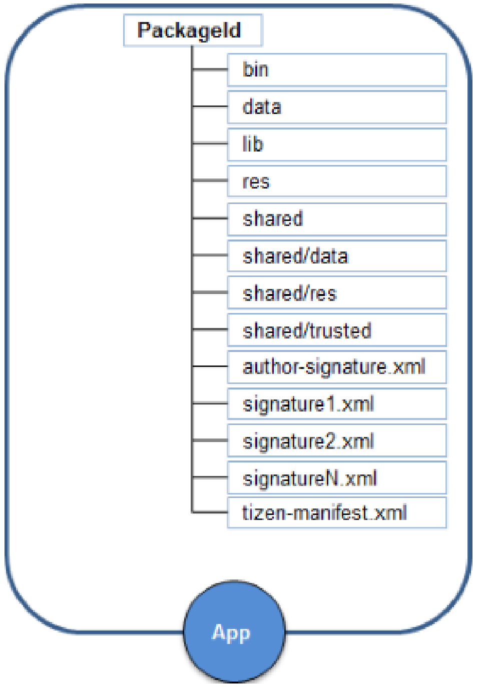
❒ Application Directory Policy

- The Tizen platform uses the underlying Linux file system
- Native applications can access the file system using Native APIs and opensource libraries such as eglibc, glib, and so on


The `tizen-manifest.xml` file and signature files are located in the application root directory.



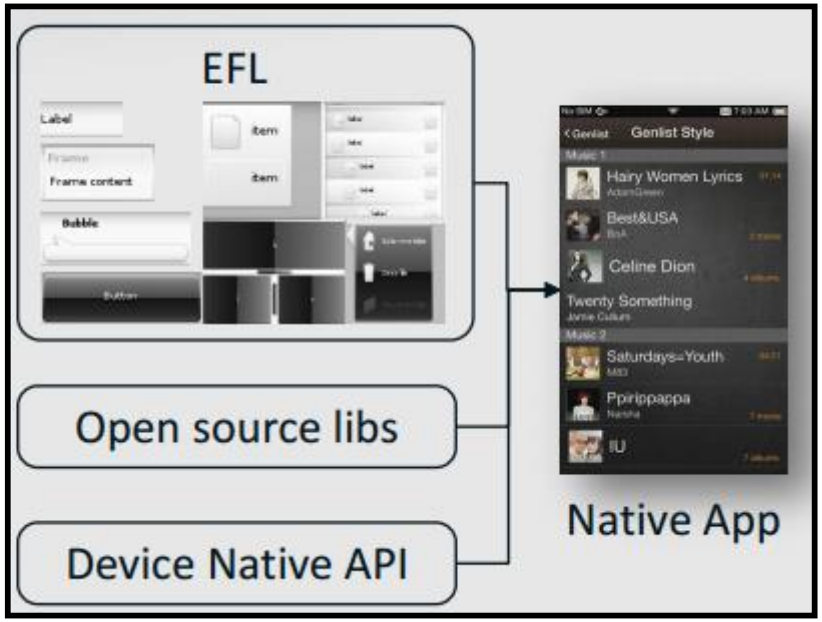
- `bin` : Executable binary path
- `lib` : Library path
- `res` : Resource path
- `data` : The application's own directory (read or write). no initial data
- `shared/` : For sharing with other applications



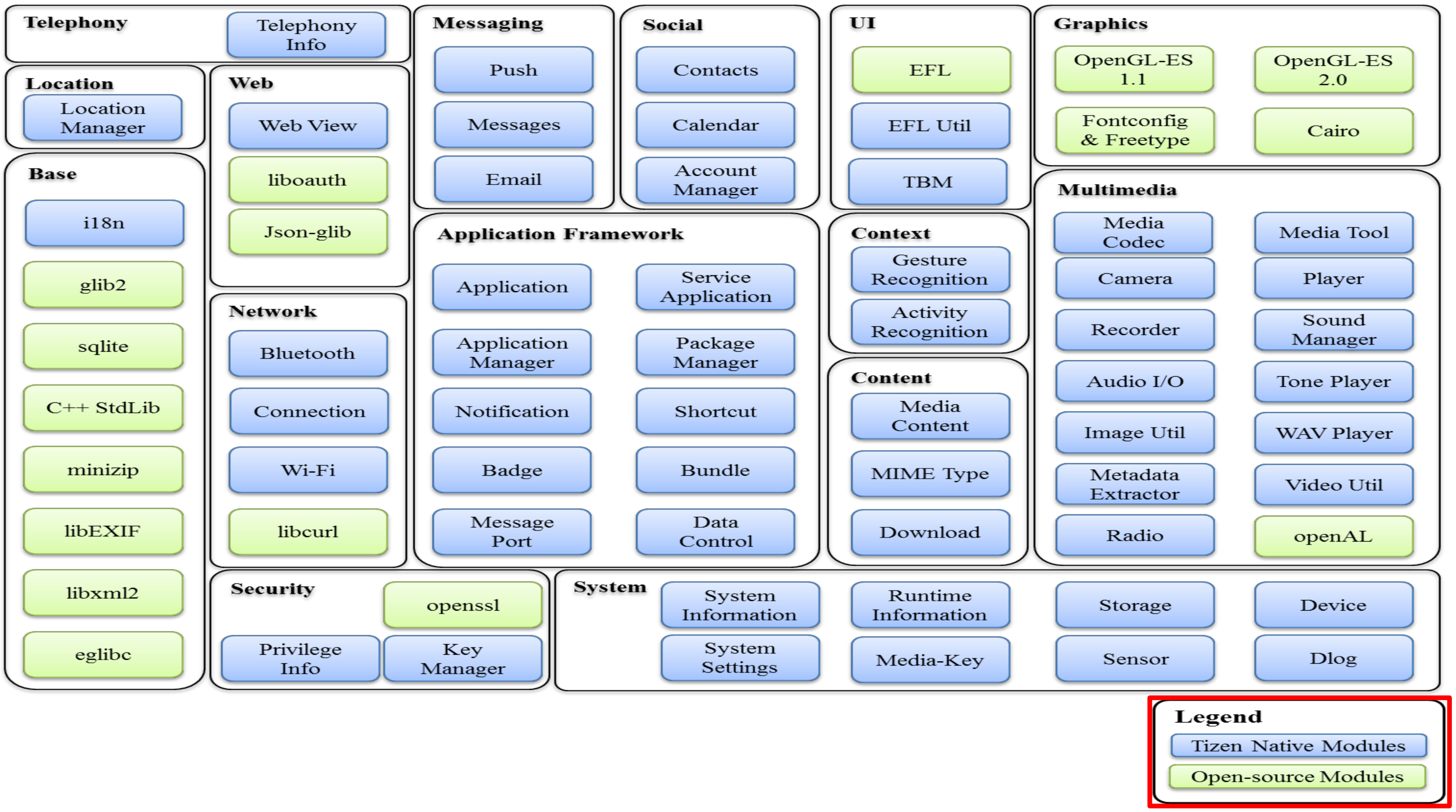
Tizen Native API

Tizen Native API

- ❒ Mature
  - Technology was already in Tizen since 1.0
  - Now in SDK and Compliance for 3rd party developers
- ❒ Powerful Graphics
  - Powered by Enlightenment Foundation Libraries(EFL)
  - High performance, scalable, customizable styles
- ❒ Lightweight
  - Light-weight enough to fit in every Tizen Profile
- ❒ More open source libraries
  - EFL, Sqlite, openssl, Curl, json-glib, libexif, etc



Tizen Native API Layout



Native API – Application Framework/Base

- Application Framework/Base
  - Application, Service Application
    - Managing the main event loop of an application or background application, managing application state changes, launching other applications using the application name, URI, or MIME type
  - Package manager
    - Storing and retrieving information related to packages installed on the device
  - Application manager
    - Information about applications
  - Notification
    - Managing notifications
  - Message-port
    - Passing messages between applications

Native API – Application Framework/Base

- Application Framework/Base
  - Message-port
    - Passing messages between applications
  - Bundle
    - Simple string-based dictionary ADT
  - Data Control
    - Exchanging specific data between applications
  - I18n
    - Flexible generation of number or date format patterns, formatting and parsing dates/number for any locale

Native API – System/Security

- System/Security
  - Sensor, Device
    - Interfaces for accessing devices such as sensors, USB, MMC, battery, CPU, and display
  - System Information, Runtime Information
    - Getting information about the device
  - System Settings
    - Getting system settings containing miscellaneous system preference
  - Dlog
    - Sending log output for debug activities

Native API – System/Security

- System/Security
  - Storage
    - Getting information about storage
  - Key-manager
    - Providing a secure repository protected by user's passwords for keys, certificates, and sensitive data
  - Privilege-Info
    - Retrieving and displaying privilege information

Native API – Network/Location

- Network/Location
  - Location Manager
    - Position information, satellite, GPS status
  - Connection
    - Managing modem data connections
  - Bluetooth
    - Managing Bluetooth devices
  - Wi-Fi
    - Managing Wi-Fi and monitoring the state of Wi-Fi

Native API – Multimedia

- Multimedia
  - Image Util
    - Encoding, decoding, and transforming images
  - Video Util
    - Transcoding a media file
  - Audio I/O
    - Recording from the audio device and playing raw audio data
  - Player
    - Playing multimedia contents from a file, network, and memory
  - Tone Player, Wav Player
    - Playing the tone and Waveform audio files
  - Camera
    - Controlling a camera device

Native API – Multimedia

- Multimedia
  - Recorder
    - Recording audio and video
  - Radio
    - Accessing Radio
  - Metadata-extractor
    - Extracting meta data from an input media file
  - Media-codec
    - Directly accessing media codes on the device
  - Media tool
    - Handling AV packet buffer for interworking between multimedia framework modules

Native API – Content/Context/Social

- Content/Context/Social
  - Media Content
    - Managing information about media files
  - Download
    - Downloading the contents of a URL to the storage asynchronously
  - Mime-type
    - Mapping MIME types to file extensions and vice versa
  - Activity Recognition, Gesture Recognition
    - Controlling information of the user and device including motions, activities

Native API – Content/Context/Social

- Content/Context/Social
  - Account Manager
    - Managing account information on the device
  - Calendar
    - Managing calendar events and accessing calendar database
  - Contacts
    - Managing contacts and contact groups and accessing contact database

Native API – UIX/Web

- UIX/Web
  - TBM Surface
    - Providing surface for Tizen
  - EFL-util
    - Getting and setting the priority order of the notification window
  - WebView
    - Displaying and controlling Web pages, such as browsing, tracking browsing history, and downloading Web content

Tizen Supported Open Source Libraries

Lib	Why we need to open this library?
EFL	EFL is the fundamental set of libraries underlying the Native API
libEXIF	Exif is an image file format used by camera and scanner devices (extends existing formats such as jpeg and tiff). Many Tizen devices have a camera and emit this format, libexif allows decoding
Json-glib	Json-glib is a library for serializing and deserializing Javascript Object Notation (JSON) using Glib and Gobject data types.
Eglibc	Standard C library, needs to be available to programs written in ISO C language
Glib	Application building blocks which add data types and other programming facilities for C-language programs
Curl	A client-side URL transfer library supporting http, https, ftp, file URIs and many more protocols. Allows applications to perform url-related activities without having to involve a web browser
libXML2	Library for parsing xml documents
Fontconfig	Font-handling library to let applications find a font or a closely matching font
Freetype	Text-rendering library
Minizip	Lightweight library building on top of zlib for processing files in the zip format
Sqlite	Implements a lightweight sql database within a library, widely used for embedded client-local storage.
Cairo	Library for 2-D vector graphics drawing
openssl	Library implementation of secure sockets layer (ssl) and transport layer security (tls) to enable secure internet Communications
OpenAL	Audio API designed for efficient rendering of 3-D positional audio.
OpenGL ES	library for rendering 3-D and 2-D graphics in embedded systems
C++ Standard Library	Standard C library, needs to be available programs written in ISO C++ language

※If you want to add another open-source, you can add your package.

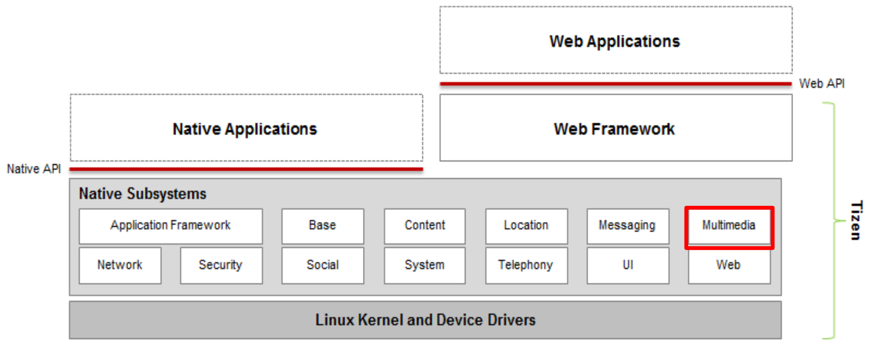
EFL Overview

- EFL(Enlightenment Foundation Libraries)
  - A set of free and open source graphics libraries that grew out of the development of the Enlightenment window manager and Wayland compositor
  - GUI Application development toolkit
  - Window System based on X11
  - Provides a set of libraries for adding common GUI widgets
  - Handling and routing input, managing data, communications and the main-loop
- Enlightenment Open source Project (http://www.enlightenment.org)
  - A whole suite of libraries to help create beautiful user interfaces with much less work

Example of
Tizen Native Application - CAMERA

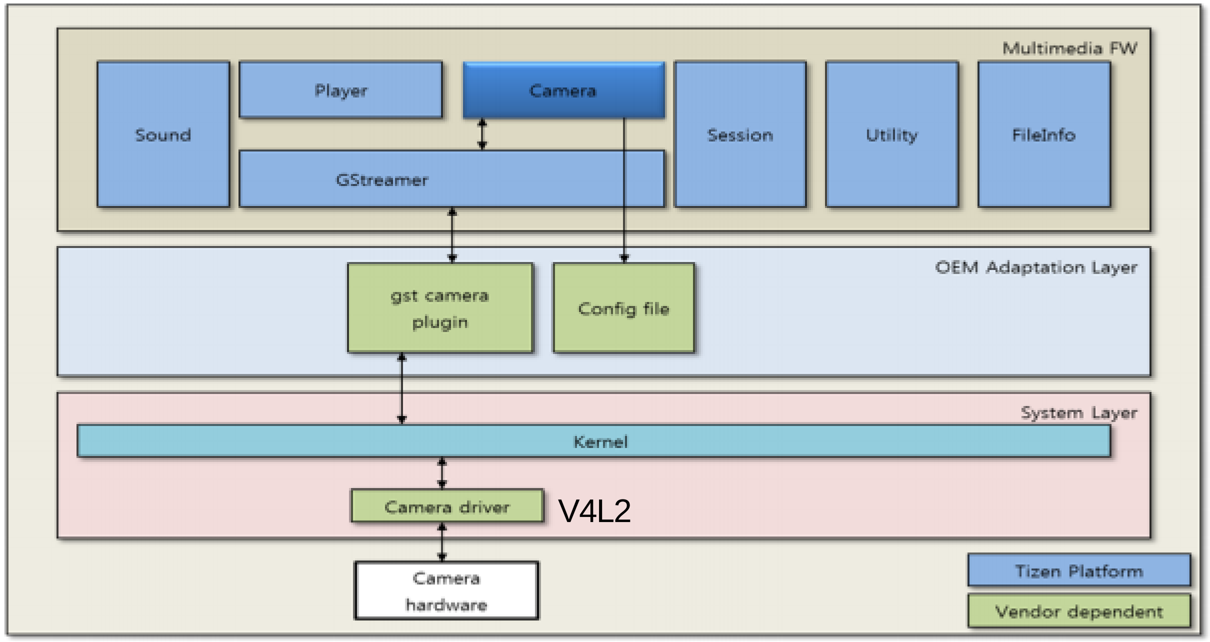
Tizen Camera Framework

Figure: Tizen architecture



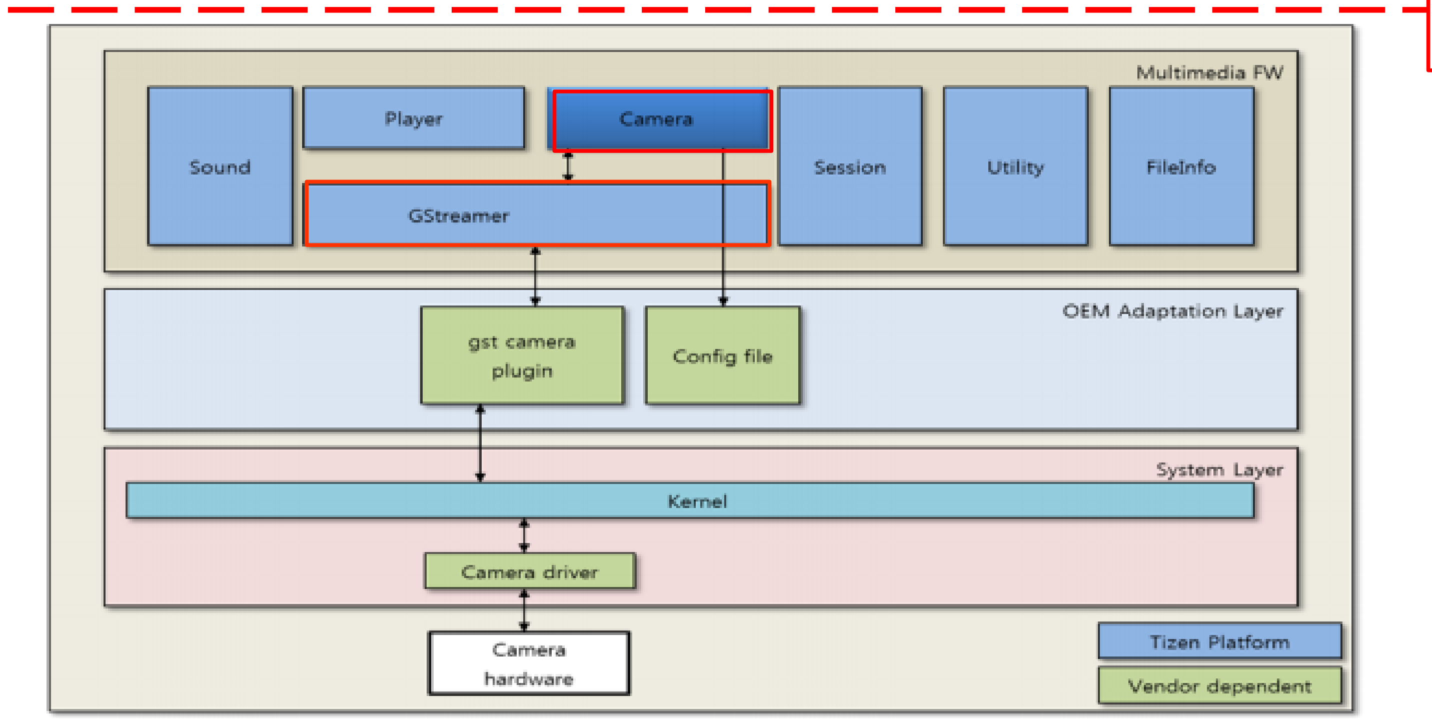
- The Multimedia camera framework controls the camera plugin of GStreamer to capture camera data from the device

Camera in the Multimedia Framework



Tizen Camera Framework

Camera API



Tizen Camera Framework

❒ Tizen Camera Framework

- Camera : A Camera Library in Tizen C API
  - Native application developer only focus on this API for controlling camera
    - ✓ Download from Tizen git : framework/api/camera
- Libmm-camcorder : Camera and recorder development library
  - Tizen Multimedia Framework for camera and recorder
    - ✓ Download from Tizen git : framework/multimedia/libmm-camcorder
- Gstreamer : open source multimedia framework
  - Streaming media framwork, based on graphs of filters which operates on media data.
    - ✓ Download from Tizen git : framework/multimedia/gstreamer

Tizen Camera API

❐ Initializing the Camera

- To use the functions and data types of the Camera API, include the <camera.h>header file in application

```
#include <camera.h>
```

❐ Creating the Camera Handle

- Create the camera handle using the camera_create() function

```
int error_code = 0;

// Create the camera handle
error_code = camera_create(CAMERA_DEVICE_CAMERA0, &cam_data.g_camera);
if (error_code == CAMERA_ERROR_NONE)
{
   dlog_print(DLOG_INFO, LOG_TAG, "error code = %d", error_code);
}
else
{
   dlog_print(DLOG_ERROR, LOG_TAG, "error code = %d", error_code);
}
```

Tizen Camera API

- Check the current state of the camera
  - To use the camera_get_state() function, if the state is not CAMERA_STATE_CREATED, reinitialize the camera

```
camera_state_e state;

// Check the camera state after creating the camera
error_code = camera_get_state(cam_data.g_camera, &state);
```

- Configuring the Camera
  - Set the image quality using the camera_attr_set_image_quality() function

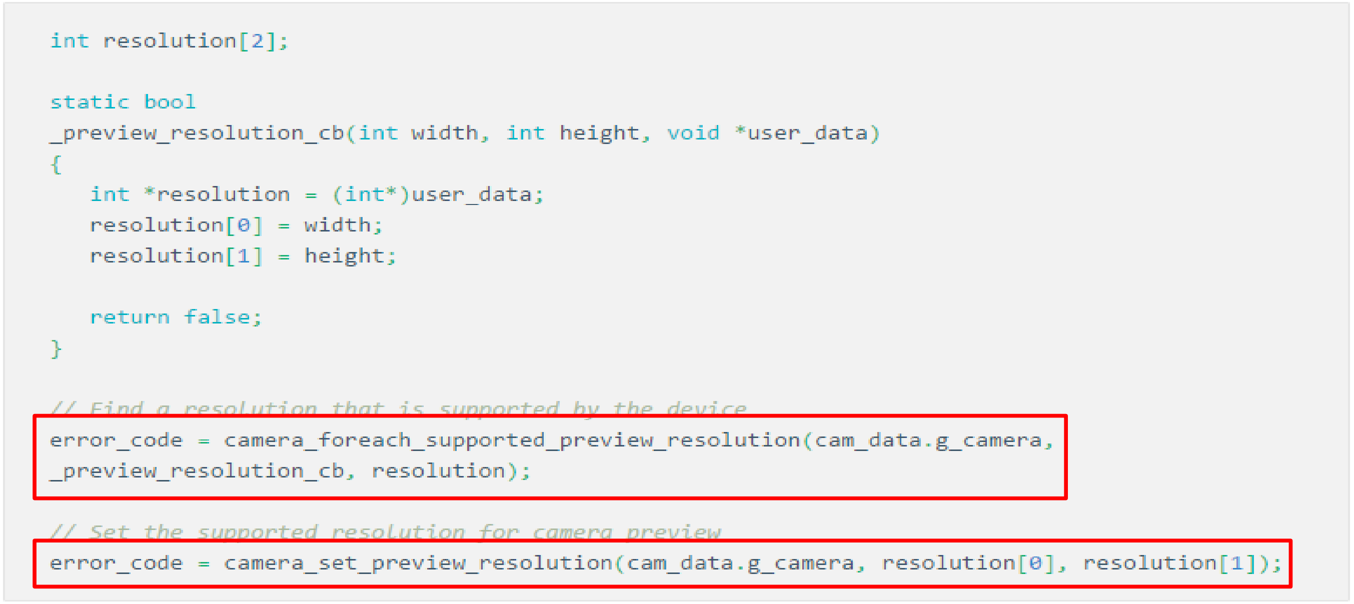
```
error_code = camera_attr_set_image_quality(cam_data.g_camera, 100);
```

  - Set Display for showing preview images by using camera_set_display() function

```
error_code = camera_set_display(cam_data.g_camera, CAMERA_DISPLAY_TYPE_OVERLAY,
GET_DISPLAY(cam_data.win));
```

Tizen Camera API

- Set the camera preview resolution
  - You must call this function before previewing
    - To use the camera_set_preview_resolution() function
  - To find out which resolutions can be set for the camera preview on a specific device
    - use the camera_foreach_supported_preview_resolution() function



```
int resolution[2];

static bool
_preview_resolution_cb(int width, int height, void *user_data)
{
   int *resolution = (int*)user_data;
   resolution[0] = width;
   resolution[1] = height;

   return false;
}

// Find a resolution that is supported by the device
error_code = camera_foreach_supported_preview_resolution(cam_data.g_camera,
_preview_resolution_cb, resolution);

// Set the supported resolution for camera preview
error_code = camera_set_preview_resolution(cam_data.g_camera, resolution[0], resolution[1]);
```

Tizen Camera API

- Set the camera capture format
  - To use the camera_set_capture_format() function
  - The camera_pixel_format_e enumeration defines the available capture formats

```
error_code = camera_set_capture_format(cam_data.g_camera, CAMERA_PIXEL_FORMAT_JPEG);
```

- Set Auto-focus Callbacks
  - To Receive notifications about auto-focus state changes, register a callback using the camera_set_focus_changed_cb() function
  - The callback is invoked every time the auto-focus state changes

```
error_code = camera_set_focus_changed_cb(cam_data.g_camera, _camera_focus_cb, NULL);
```

Tizen Camera API

- Set Preview Callback
  - To receive notifications about newly previewed frames, register a callback using the camera_set_preview_cb()function
  - The callback is invoked once per frame during a preview

```
error_code = camera_set_preview_cb(cam_data.g_camera, _camera_preview_cb, NULL);
```

  - The following example code implements

```
static void
_camera_preview_cb(camera_preview_data_s *frame, void *user_data)
{
    int error_code = 0;

    if (g_enable_focus == true)
    {
        error_code = camera_start_focusing(cam_data.g_camera, true);

        if (error_code == CAMERA_ERROR_NOT_SUPPORTED)
        {
            error_code = camera_start_focusing(cam_data.g_camera, false);
        }

        g_enable_focus = false;
    }
}
```

Tizen Camera API

❒ Taking a Photo

- After initializing the camera, start the camera preview using the camera_start_preview() function

```
error_code = camera_start_preview(cam_data.g_camera);
```

- The camera preview draws preview frames on the screen and allows to capture frames as still images
  - To handle the camera preview, the application uses the camera preview callback
- After start preview, start capturing of still image using camera_start_capture() function

```
    if (state == CAMERA_FOCUS_STATE_FOCUSED && g_enable_shot == true)
    {
        //  Start capturing
        error_code = camera_start_capture(cam_data.g_camera, _camera_capturing_cb,
_camera_completed_cb, NULL);

        g_enable_shot = false;
    }
```

Tizen Camera API

❒ Process of Capturing by callback function(1)

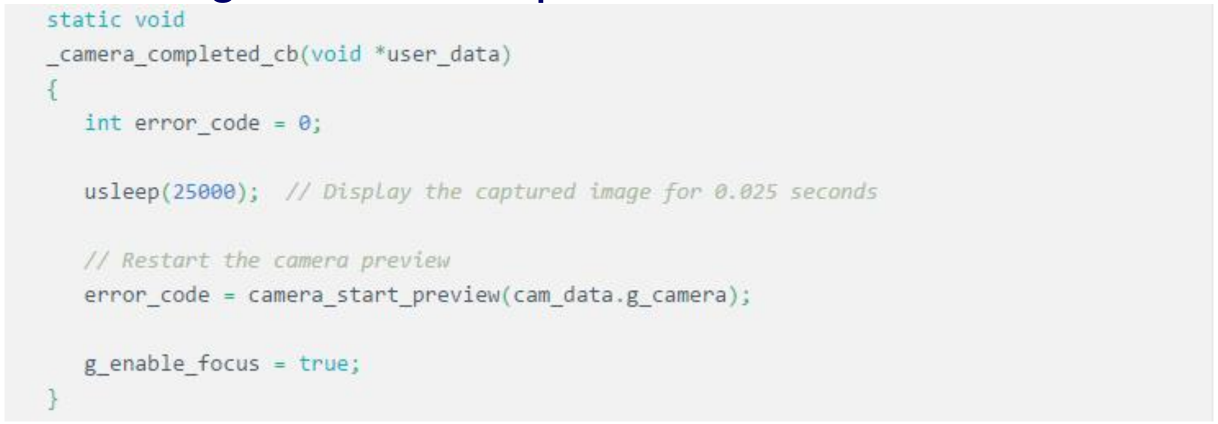
- To handle the capturing process, the application calls callback function at camera_start_capture function(ex. camera_capturing_cb)
- This callback is invoked once for each captured frame. The image is saved in the format set by the camera_set_capture_format() function
- The following example code implements the _camera_capturing cb() callback, which saves the captured frame as a JPEG image

```
static void
_camera_capturing_cb(camera_image_data_s* image, camera_image_data_s* postview,
camera_image_data_s* thumbnail, void *user_data)
{
   dlog_print(DLOG_DEBUG, LOG_TAG, "Writing image to file");
   FILE *file = fopen(g_fname, "w+");

   if (image->data != NULL)
   {
      fwrite(image->data, 1, image->size, file);
   }
   fclose(file);
}
```

Tizen Camera API

❒ Process of Capturing by callback function(2)

- To receive a notification when image has been captured , implement camera_capture_completed_cb() callback
- This callback is invoked after camera_capturing_cb() callback completes, and is used for notification and for restarting the camera preview
- The following example code implements the _camera_complete_cb() callback, which waits 0.025 seconds before restarting the camera preview



```
static void
_camera_completed_cb(void *user_data)
{
    int error_code = 0;

    usleep(25000);  // Display the captured image for 0.025 seconds

    // Restart the camera preview
    error_code = camera_start_preview(cam_data.g_camera);

    g_enable_focus = true;
}
```

Tizen Camera API

❒ Set Camera Attributes

- Preview frame rates – need to set before starting preview
- Using camera_attr_set_preview_fps() function

```
error_code = camera_attr_set_preview_fps(cam_data.g_camera, CAMERA_ATTR_FPS_AUTO);
```

- Zoom Attribute - Retrive the range of available zoom level values using the camera_attr_get_zoom_range() function
  - Set camera zoom by camera_attr_set_zoom() function

```
int min, max;

error_code = camera_attr_get_zoom_range(cam_data.g_camera, &min, &max);

error_code = camera_attr_set_zoom(cam_data.g_camera, min);
```

- Brightness Attribute - Retrieve the range of available brightness level values using the camera_attr_get_brightness_range() function, and the current brightness level using the camera_attr_get_brightness() function

Tizen Camera API

❒ Releasing Resources

- Stop the camera preview using camera_stop_preview() function

```
error_code = camera_stop_preview(cam_data.g_camera);
```

- Destory the camera handle and release all its resources using the camera_destory() function

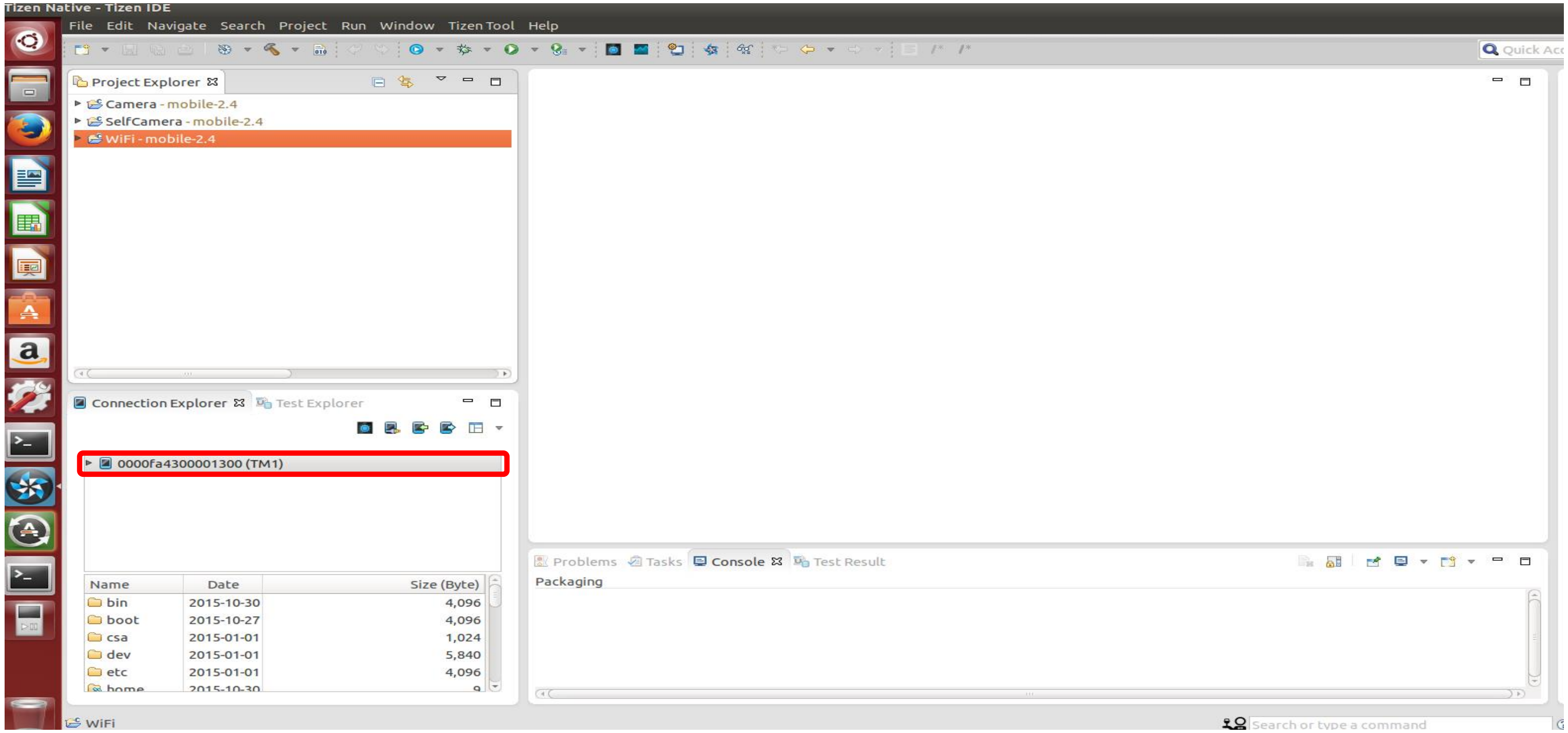
```
error_code = camera_destroy(cam_data.g_camera);
```

LAB : Tizen Native Sample Application - Camera

Sample Native Application – Camera

❒ Connect mobile phone at Tizen IDE

- Connect mobile phone to PC with usb cable
- Run Tizen IDE -> Check mobile Device at Connection Explorer



Camera - mobile-2.4
SelfCamera - mobile-2.4
WiFi - mobile-2.4

0000fa4300001300 (TM1)

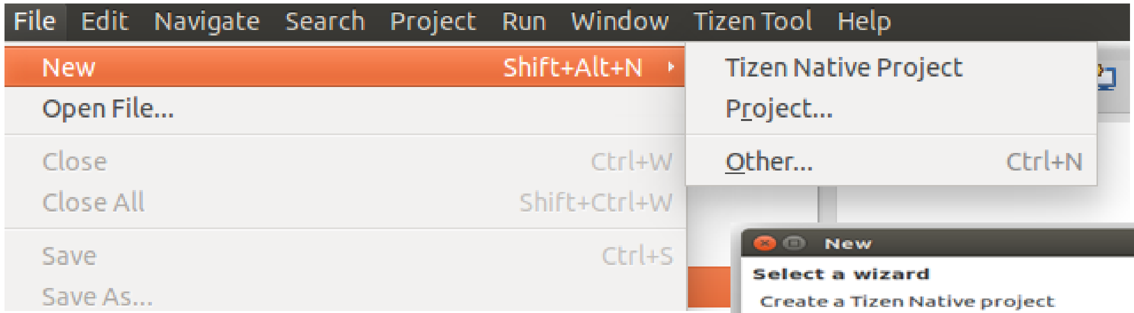
Name	Date	Size (Byte)
bin	2015-10-30	4,096
boot	2015-10-27	4,096
csa	2015-01-01	1,024
dev	2015-01-01	5,840
etc	2015-01-01	4,096
home	2015-10-30	9

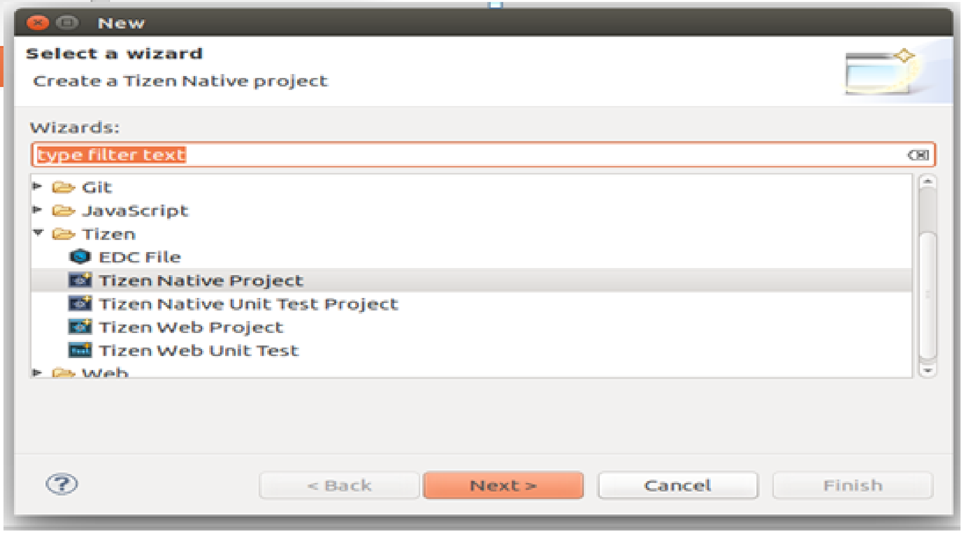
Packaging

Sample Native Application – Camera

❒ Create New Project

- Main Menu -> [File > New > Other…] select
- Open Tizen folder -> Select Tizen Native Project at List

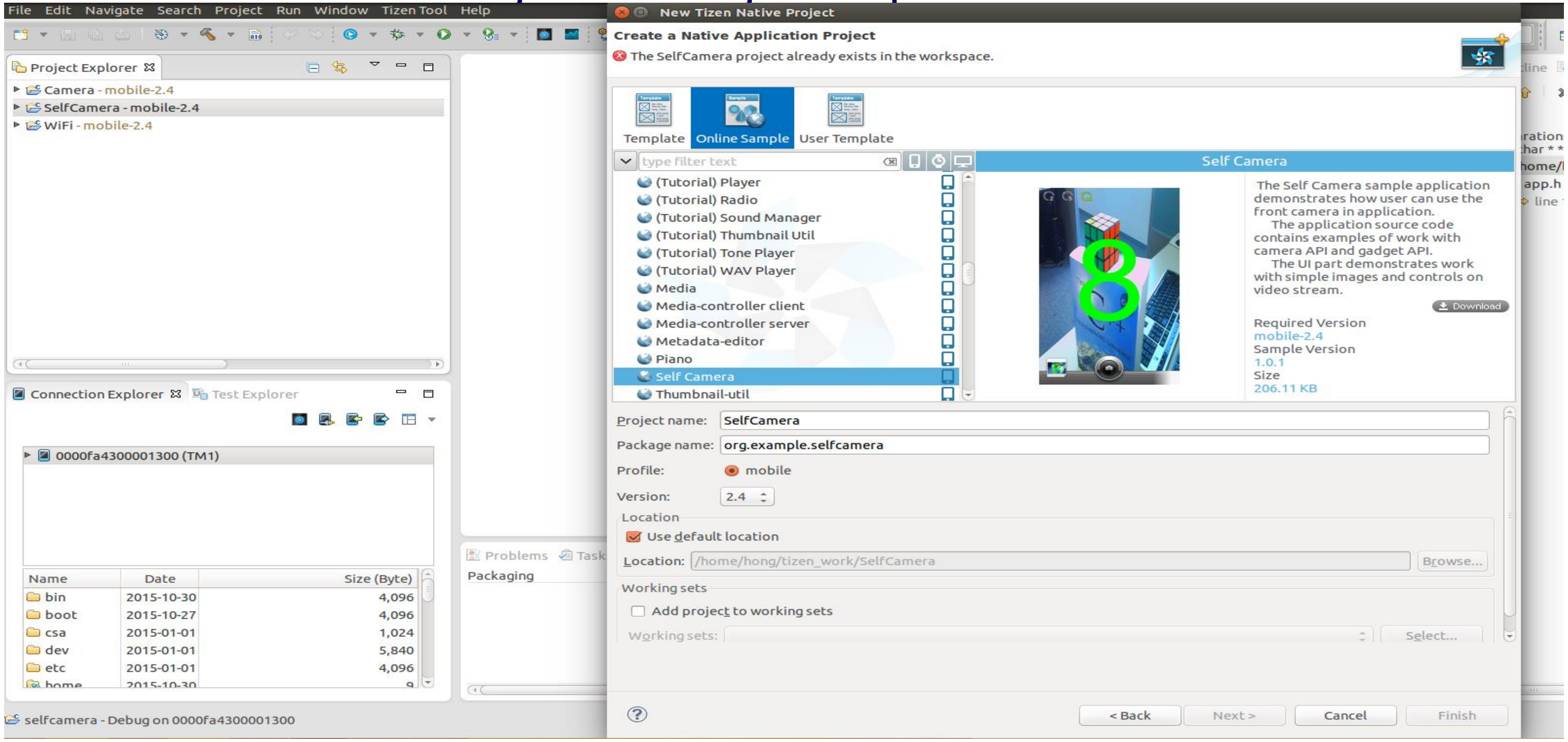




Sample Native Application – Camera

❒ Sample Native Application

- Online Sample -> Select Camera
- Check Camera Project at Project Explorer



Name	Date	Size (Byte)
bin	2015-10-30	4,096
boot	2015-10-27	4,096
csa	2015-01-01	1,024
dev	2015-01-01	5,840
etc	2015-01-01	4,096
home	2015-10-30	9

New Tizen Native Project

Create a Native Application Project

The SelfCamera project already exists in the workspace.

(Tutorial) Player
(Tutorial) Radio
(Tutorial) Sound Manager
(Tutorial) Thumbnail Util
(Tutorial) Tone Player
(Tutorial) WAV Player
Media
Media-controller client
Media-controller server
Metadata-editor
Piano
Self Camera
Thumbnail-util

Self Camera

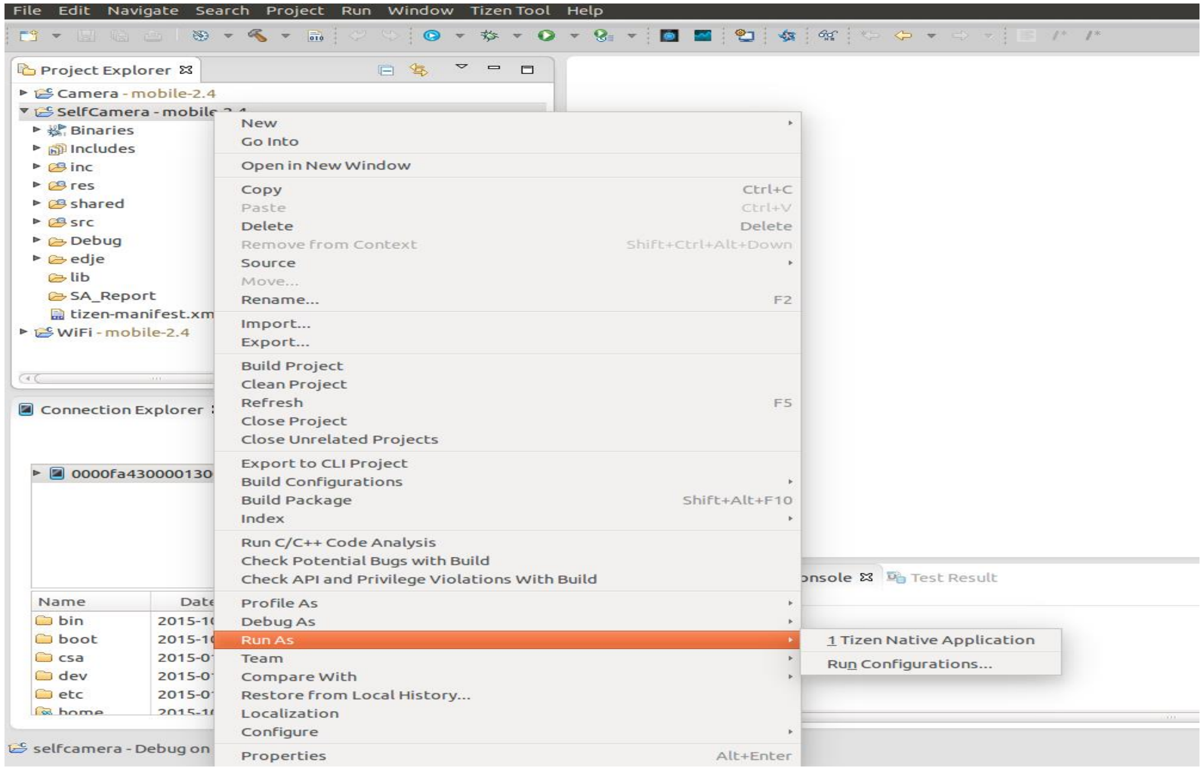
The Self Camera sample application demonstrates how user can use the front camera in application.
The application source code contains examples of work with camera API and gadget API.
The UI part demonstrates work with simple images and controls on video stream.

Required Version mobile-2.4
Sample Version 1.0.1
Size 206.11 KB

Project name: SelfCamera
Package name: org.example.selfcamera
Profile: mobile
Version: 2.4
Location
[✓] Use default location
Location: /home/hong/tizen_work/SelfCamera Browse...
Working sets
[ ] Add project to working sets
Working sets: Select...

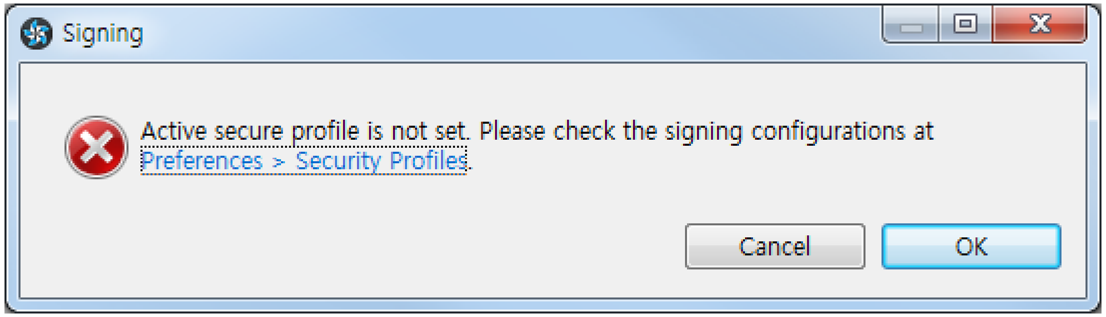
Sample Native Application – Camera

- ❒ Self Camera Build : Project -> Build Project
- ❒ Select mobile device at Connection Explorer
- ❒ Right Click at Project Explorer -> Run As -> Tizen Native Application



Sample Native Application – Camera

- ❒ If Secure Profile Warning popup is occurred, you need to generate a new certificate

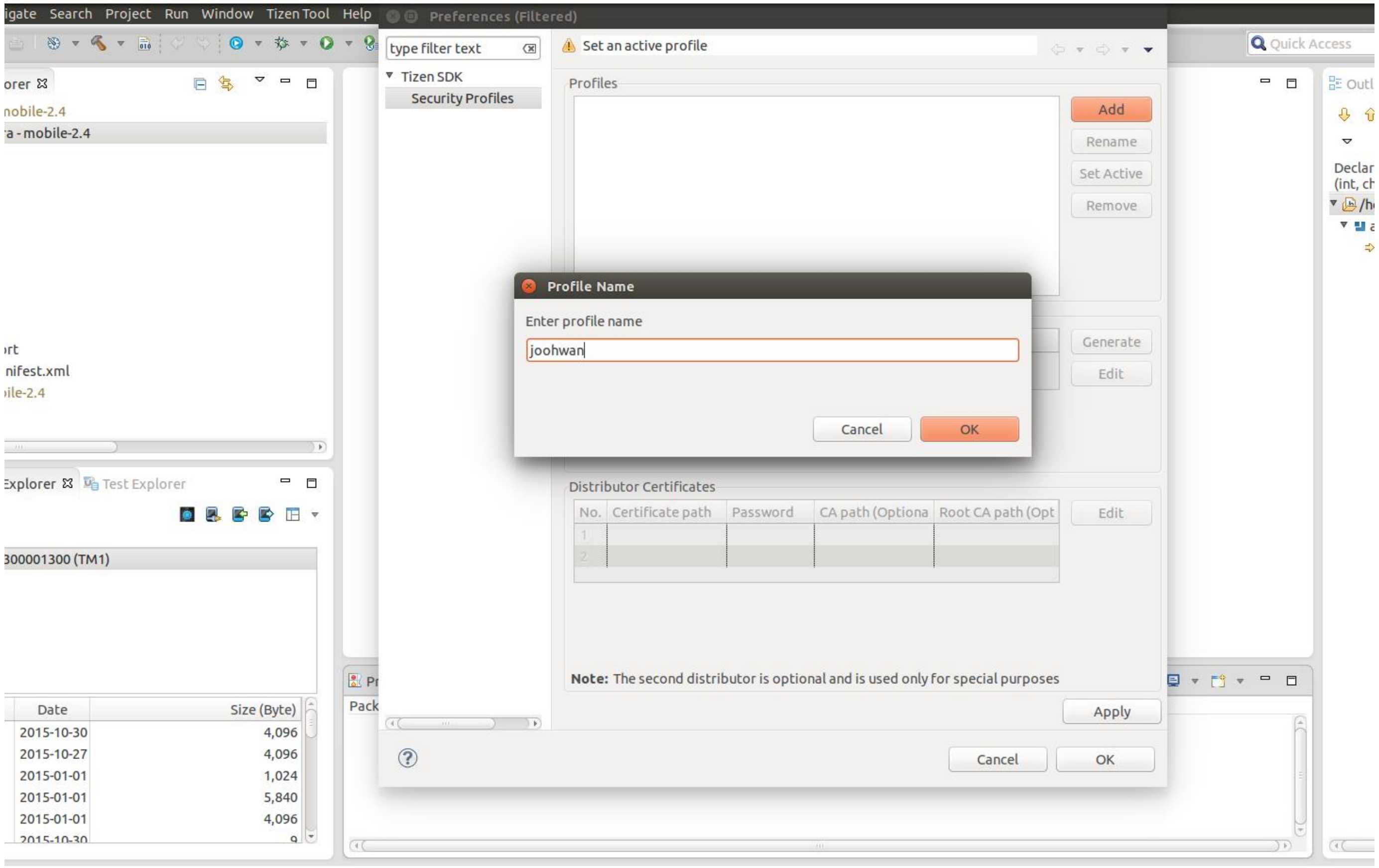


Active secure profile is not set. Please check the signing configurations at Preferences > Security Profiles.

- ❒ Select "Preferences > Security Profiles" Link

Sample Native Application – Camera

- Enter Profile name -> add more information -> Generate
- Enter mandatory information * list at pop-up window -> OK



Set an active profile

Profiles

Profile Name

Enter profile name

joohwan

Cancel OK

Distributor Certificates

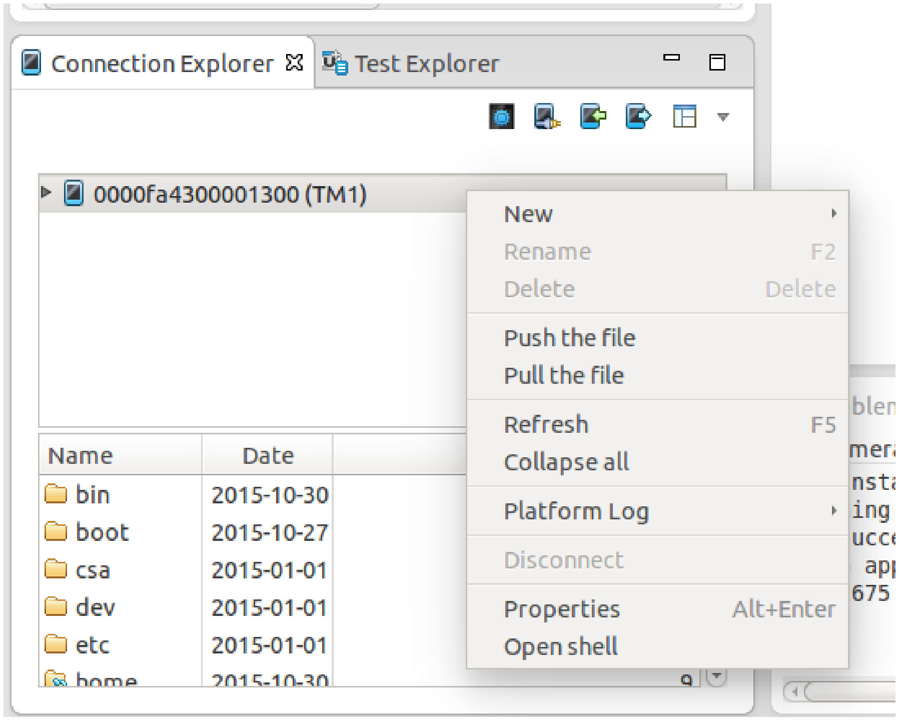
No.	Certificate path	Password	CA path (Optiona	Root CA path (Opt
1				
2				

Note: The second distributor is optional and is used only for special purposes

Date	Size (Byte)
2015-10-30	4,096
2015-10-27	4,096
2015-01-01	1,024
2015-01-01	5,840
2015-01-01	4,096
2015-10-30	9

Sample Native Application – Camera

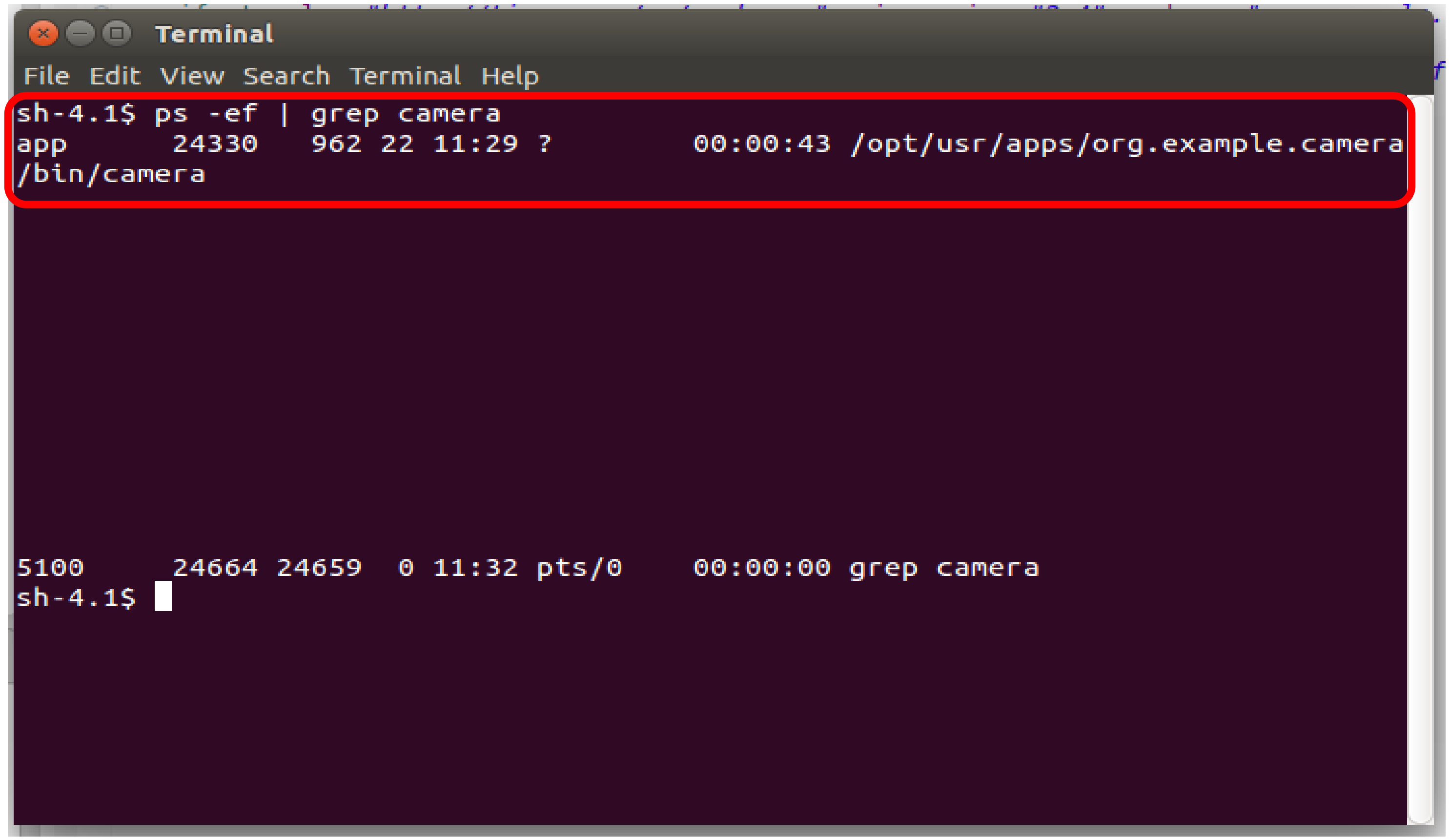
- Check Self camera application is installed at mobile phone
- Open Target Device Shell
  - Right Click at Connection Explorer and select "Open shell"



Name	Date
bin	2015-10-30
boot	2015-10-27
csa	2015-01-01
dev	2015-01-01
etc	2015-01-01
home	2015-10-30

Sample Native Application – Camera

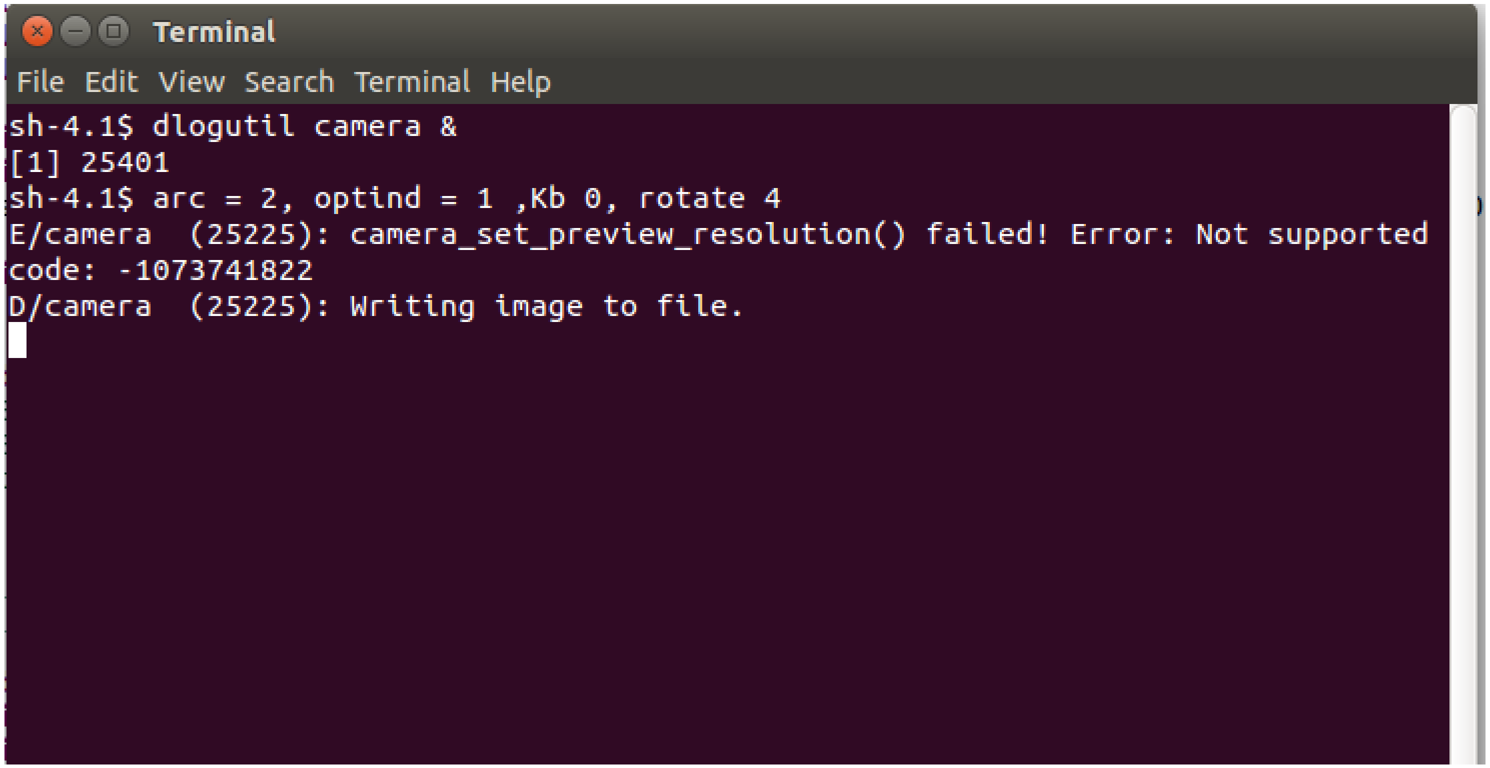
❒ Enter “ps –ef | grep camera” at console and check application is executed or not



```
sh-4.1$ ps -ef | grep camera
app      24330   962 22 11:29 ?        00:00:43 /opt/usr/apps/org.example.camera
/bin/camera








5100     24664 24659  0 11:32 pts/0    00:00:00 grep camera
sh-4.1$ 
```

Sample Native Application – Camera

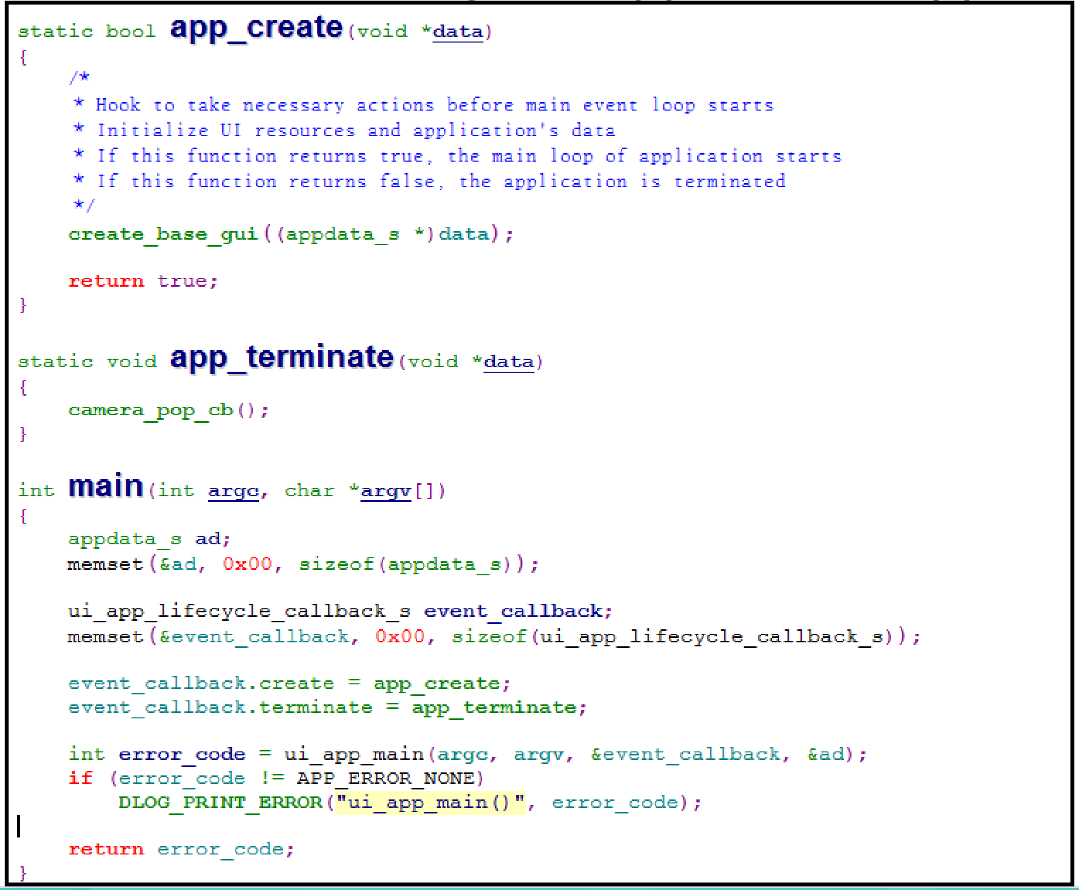
- Check Self Camera Application log at console
  - Using dlogutil for checking log (find log tag at logger.h or main.h)
  - Enter 'dlogutil camera &' at console and then log is printed out



```
sh-4.1$ dlogutil camera &
[1] 25401
sh-4.1$ arc = 2, optind = 1 ,Kb 0, rotate 4
E/camera  (25225): camera_set_preview_resolution() failed! Error: Not supported
code: -1073741822
D/camera  (25225): Writing image to file.
```

Sample Native Application – Camera

- Code Review – main.c
  - Main function - register app_create, app_terminate call back function



```
static bool app_create(void *data)
{
    /*
     * Hook to take necessary actions before main event loop starts
     * Initialize UI resources and application's data
     * If this function returns true, the main loop of application starts
     * If this function returns false, the application is terminated
     */
    create_base_gui((appdata_s *)data);

    return true;
}

static void app_terminate(void *data)
{
    camera_pop_cb();
}

int main(int argc, char *argv[])
{
    appdata_s ad;
    memset(&ad, 0x00, sizeof(appdata_s));

    ui_app_lifecycle_callback_s event_callback;
    memset(&event_callback, 0x00, sizeof(ui_app_lifecycle_callback_s));

    event_callback.create = app_create;
    event_callback.terminate = app_terminate;

    int error_code = ui_app_main(argc, argv, &event_callback, &ad);
    if (error_code != APP_ERROR_NONE)
        DLOG_PRINT_ERROR("ui_app_main()", error_code);

    return error_code;
}
```

Sample Native Application – Camera

- Code Review - User_callback.c
  - Camera Handler and GUI Structure

```c
typedef struct _camdata {
    camera_h g_camera;          // Camera handle
    Evas_Object *cam_display;
    Evas_Object *cam_display_box;
    Evas_Object *display;
    Evas_Object *preview_bt;
    Evas_Object *zoom_bt;
    Evas_Object *brightness_bt;
    Evas_Object *photo_bt;
    Evas_Object *toggle_bt;
    bool cam_prev;
    //int cam_dev;
} camdata;
static camdata cam_data;
```

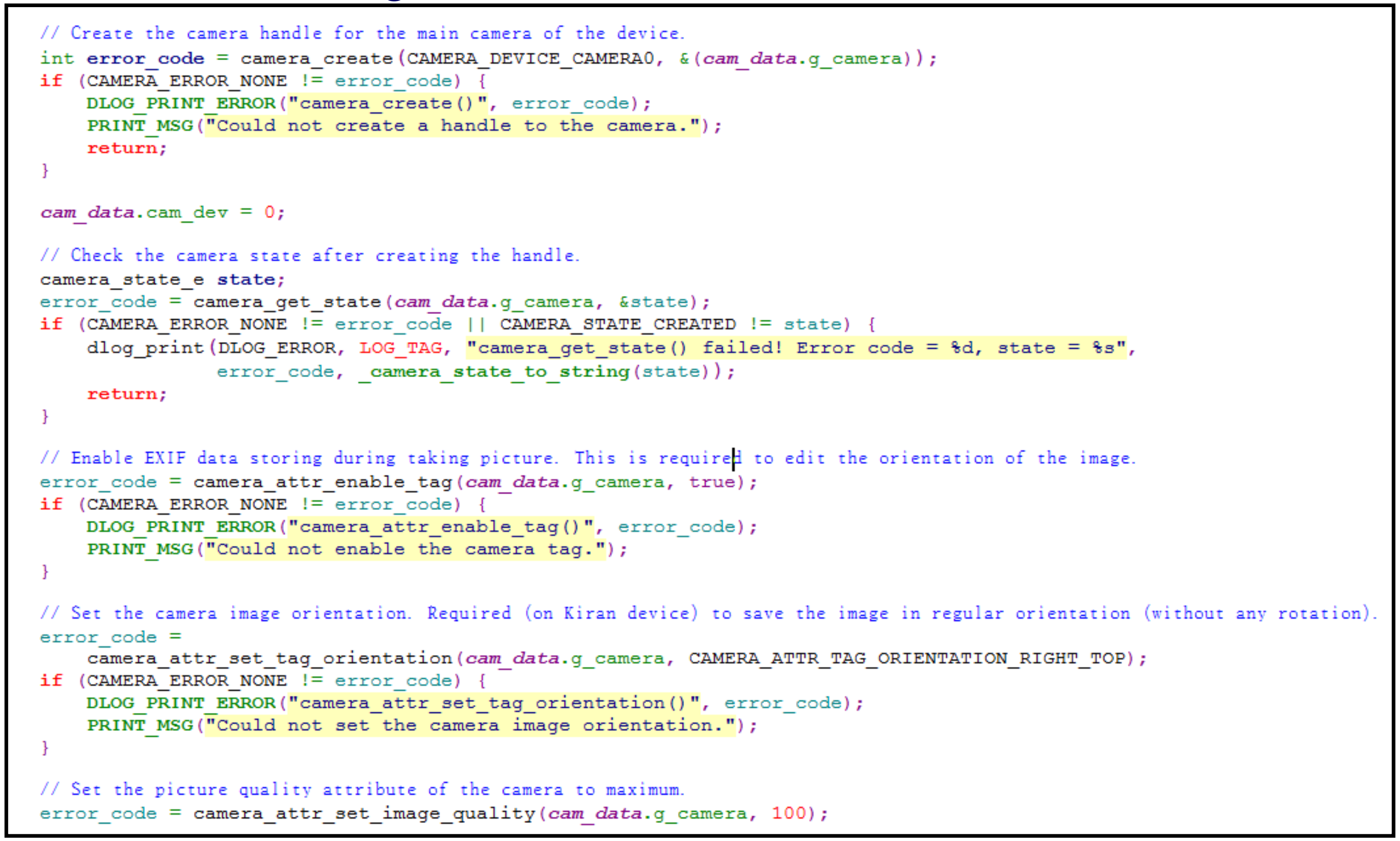
  - Create Button and button Click Callback in create _buttons _in _main _window function

```c
// Create buttons for the Camera.
cam_data.preview_bt = _new_button(ad, cam_data.display, "Start preview", __camera_cb_preview);
cam_data.zoom_bt = _new_button(ad, cam_data.display, "Zoom", __camera_cb_zoom);
cam_data.brightness_bt = _new_button(ad, cam_data.display, "Brightness", __camera_cb_bright);
cam_data.photo_bt = _new_button(ad, cam_data.display, "Take a photo", __camera_cb_photo);
```

Sample Native Application – Camera

❒ Code Review - User_callback.c

- Camera create, get state and set camera attribute

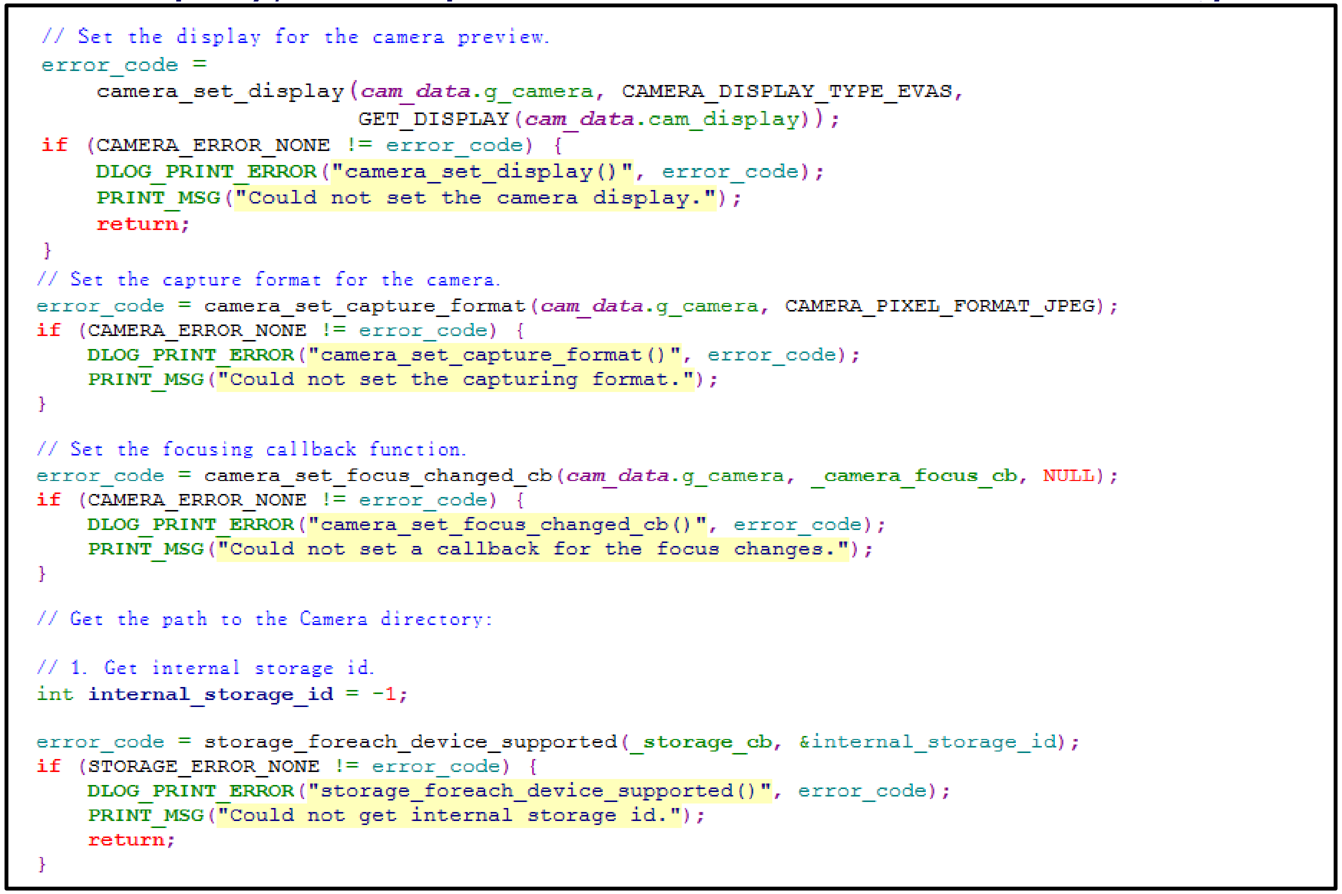


```
// Create the camera handle for the main camera of the device.
int error_code = camera_create(CAMERA_DEVICE_CAMERA0, &(cam_data.g_camera));
if (CAMERA_ERROR_NONE != error_code) {
    DLOG_PRINT_ERROR("camera_create()", error_code);
    PRINT_MSG("Could not create a handle to the camera.");
    return;
}

cam_data.cam_dev = 0;

// Check the camera state after creating the handle.
camera_state_e state;
error_code = camera_get_state(cam_data.g_camera, &state);
if (CAMERA_ERROR_NONE != error_code || CAMERA_STATE_CREATED != state) {
    dlog_print(DLOG_ERROR, LOG_TAG, "camera_get_state() failed! Error code = %d, state = %s",
               error_code, _camera_state_to_string(state));
    return;
}

// Enable EXIF data storing during taking picture. This is required to edit the orientation of the image.
error_code = camera_attr_enable_tag(cam_data.g_camera, true);
if (CAMERA_ERROR_NONE != error_code) {
    DLOG_PRINT_ERROR("camera_attr_enable_tag()", error_code);
    PRINT_MSG("Could not enable the camera tag.");
}

// Set the camera image orientation. Required (on Kiran device) to save the image in regular orientation (without any rotation).
error_code =
    camera_attr_set_tag_orientation(cam_data.g_camera, CAMERA_ATTR_TAG_ORIENTATION_RIGHT_TOP);
if (CAMERA_ERROR_NONE != error_code) {
    DLOG_PRINT_ERROR("camera_attr_set_tag_orientation()", error_code);
    PRINT_MSG("Could not set the camera image orientation.");
}

// Set the picture quality attribute of the camera to maximum.
error_code = camera_attr_set_image_quality(cam_data.g_camera, 100);
```

Sample Native Application – Camera

❒ Code Review - User_callback.c

- Set Display, set capture format and set focus change callback



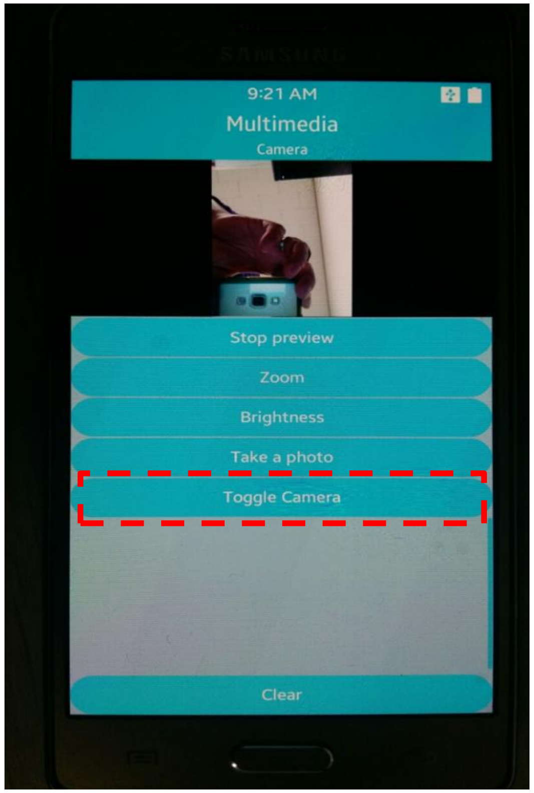
```
// Set the display for the camera preview.
error_code =
    camera_set_display(cam_data.g_camera, CAMERA_DISPLAY_TYPE_EVAS,
                       GET_DISPLAY(cam_data.cam_display));
if (CAMERA_ERROR_NONE != error_code) {
    DLOG_PRINT_ERROR("camera_set_display()", error_code);
    PRINT_MSG("Could not set the camera display.");
    return;
}
// Set the capture format for the camera.
error_code = camera_set_capture_format(cam_data.g_camera, CAMERA_PIXEL_FORMAT_JPEG);
if (CAMERA_ERROR_NONE != error_code) {
    DLOG_PRINT_ERROR("camera_set_capture_format()", error_code);
    PRINT_MSG("Could not set the capturing format.");
}

// Set the focusing callback function.
error_code = camera_set_focus_changed_cb(cam_data.g_camera, _camera_focus_cb, NULL);
if (CAMERA_ERROR_NONE != error_code) {
    DLOG_PRINT_ERROR("camera_set_focus_changed_cb()", error_code);
    PRINT_MSG("Could not set a callback for the focus changes.");
}

// Get the path to the Camera directory:

// 1. Get internal storage id.
int internal_storage_id = -1;

error_code = storage_foreach_device_supported(_storage_cb, &internal_storage_id);
if (STORAGE_ERROR_NONE != error_code) {
    DLOG_PRINT_ERROR("storage_foreach_device_supported()", error_code);
    PRINT_MSG("Could not get internal storage id.");
    return;
}
```

LAB Assignment: CAMERA SAMPLE APP - Add Camera Toggle Button

Add Camera Toggle Button – Camera

❒ Add Camera Toggle Button at Camera Sample Application

- If Toggle Button is clicked, Camera is switched between second and primary



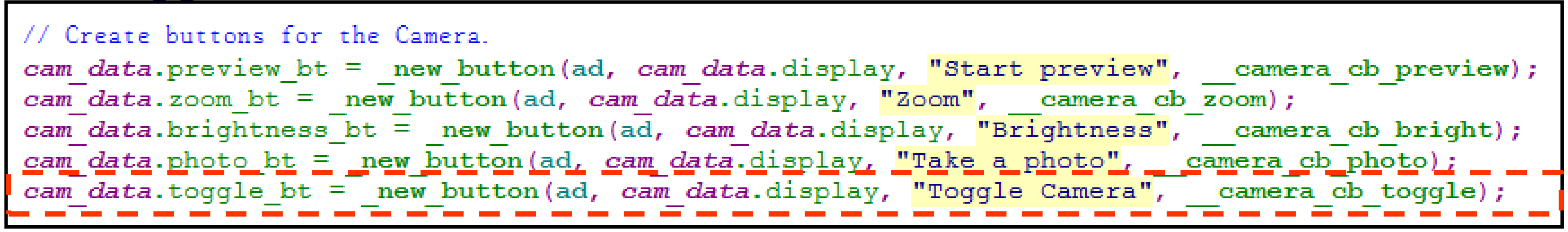
Need to DO!!!

- Add toggle button
- Set a current camera device number when camera is created
- Implement Toggle Button click call back function
- If toggle button is clicked, release and destroy current camera resource and create another camera device

Add Camera Toggle Button – Camera

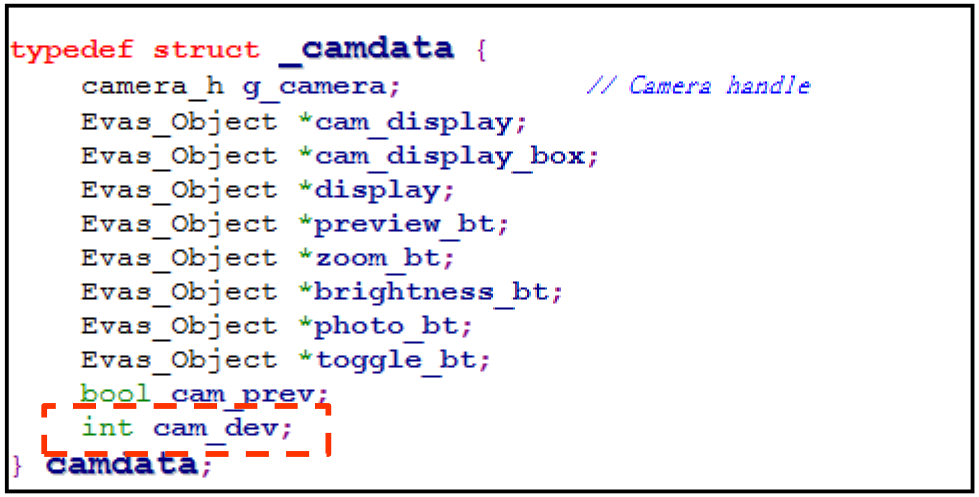
❒ Add Camera Toggle Button at Camera Sample Application

- Add toggle button



```
// Create buttons for the Camera.
cam_data.preview_bt = _new_button(ad, cam_data.display, "Start preview", __camera_cb_preview);
cam_data.zoom_bt = _new_button(ad, cam_data.display, "Zoom", __camera_cb_zoom);
cam_data.brightness_bt = _new_button(ad, cam_data.display, "Brightness", __camera_cb_bright);
cam_data.photo_bt = _new_button(ad, cam_data.display, "Take a photo", __camera_cb_photo);
cam_data.toggle_bt = _new_button(ad, cam_data.display, "Toggle Camera", __camera_cb_toggle);
```

- Add camera device number variable in camera structure



```
typedef struct _camdata {
    camera_h g_camera;          // Camera handle
    Evas_Object *cam_display;
    Evas_Object *cam_display_box;
    Evas_Object *display;
    Evas_Object *preview_bt;
    Evas_Object *zoom_bt;
    Evas_Object *brightness_bt;
    Evas_Object *photo_bt;
    Evas_Object *toggle_bt;
    bool cam_prev;
    int cam_dev;
} camdata;
```