**CRESPRIT MQTT Sample App**

**Guideline for Android**

September 23, 2014

**CRESPRIT Inc.**

**Table of Contents**

[1. Overview 4](#_Toc399250852)

[2. 배포 파일 5](#_Toc399250853)

[3. 이클립스에서의 빌드 5](#_Toc399250854)

[3.１ libdmrcp.so 5](#_Toc399250855)

[3.２ interface.jar 5](#_Toc399250856)

[4. APIs 6](#_Toc399250857)

[4.１ StartCP 6](#_Toc399250858)

[4.２ StopCP 6](#_Toc399250859)

[4.３ GetDeviceList 6](#_Toc399250860)

[4.４ Play 7](#_Toc399250861)

[4.５ Stop 8](#_Toc399250862)

[4.６ Pause 8](#_Toc399250863)

[4.７ Resume 8](#_Toc399250864)

[4.８ Seek 8](#_Toc399250865)

[4.９ GetVolume 9](#_Toc399250866)

[4.１０ SetVolume 9](#_Toc399250867)

[4.１１ GetTransportInfo 9](#_Toc399250868)

[4.１２ GetPositionInfo 10](#_Toc399250869)

[4.１３ GetCurrentMediaInfo 10](#_Toc399250870)

[5. CallBack 10](#_Toc399250871)

[6. 구현 가이드 12](#_Toc399250872)

[6.１ DMR CP Engine 실행순서 12](#_Toc399250873)

[6.２ Example Source Code 13](#_Toc399250874)

[7. Sample Application Guide 15](#_Toc399250875)

[7.１ EngineStart 15](#_Toc399250876)

[7.２ Search DMR Devices 15](#_Toc399250877)

[7.３ Radio Button 15](#_Toc399250878)

[7.４ Play 15](#_Toc399250879)

[7.５ No Selected DMR 16](#_Toc399250880)

[<표 1> Error Message 17](#_Toc399250881)

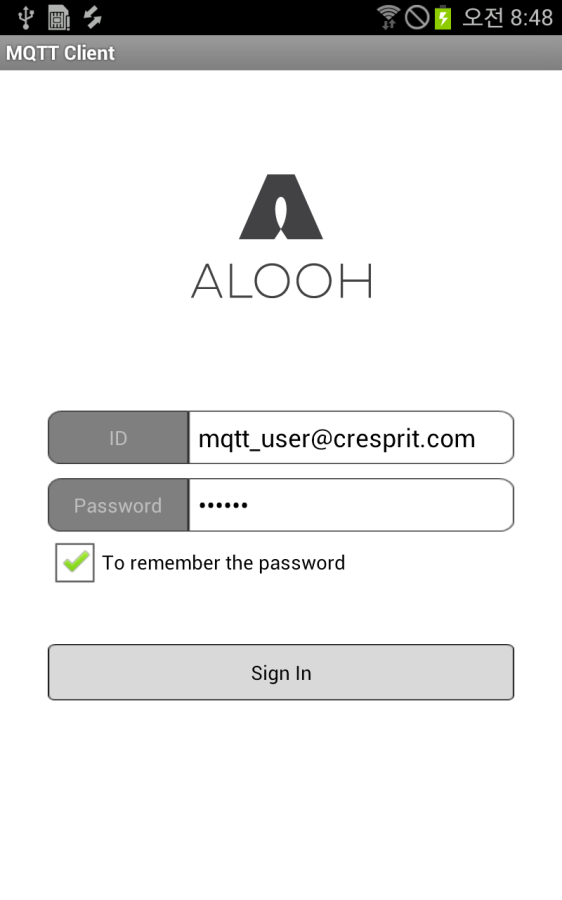
# Overview

이 문서는 Alooh MQTT 메시지 서버를 이용하여 Alooh Device로부터 전달되는 센서데이터를 모니터링 하거나 Alooh Device로 데이터를 전달하기 위한 MQTT 샘플 App에 대한 가이드를 위한 것이다. 그리고 Alooh MQTT 메시지 서버를 이용한 모든 데이터는 JSON 형태로 전달된다.

해당 MQTT 샘플 App을 이용하기 위해서는 Alooh Cloude 서버에 계정이 생성되어야 하며 안드로이드용 설정 App을 통해 Alooh Device가 등록이 되어 있어야 한다.

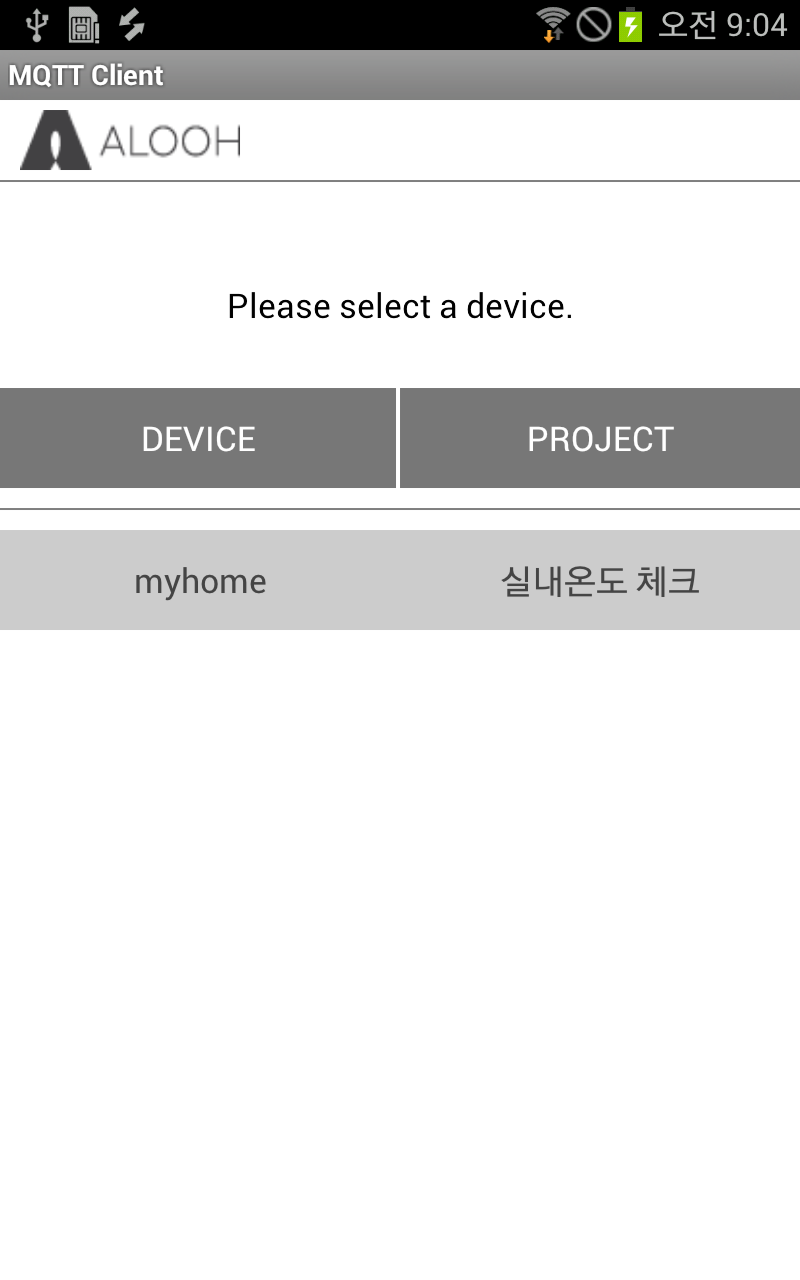
# 시나리오

1. **로그인**

* 로그인 화면은 아래와 같이 구성된다. 로그인을 위한 계정은 Alooh 홈페이지(<http://www.alooh.io/>)나 안드로이드용 Alooh Device 설정 App을 통해서 생성이 가능하다.

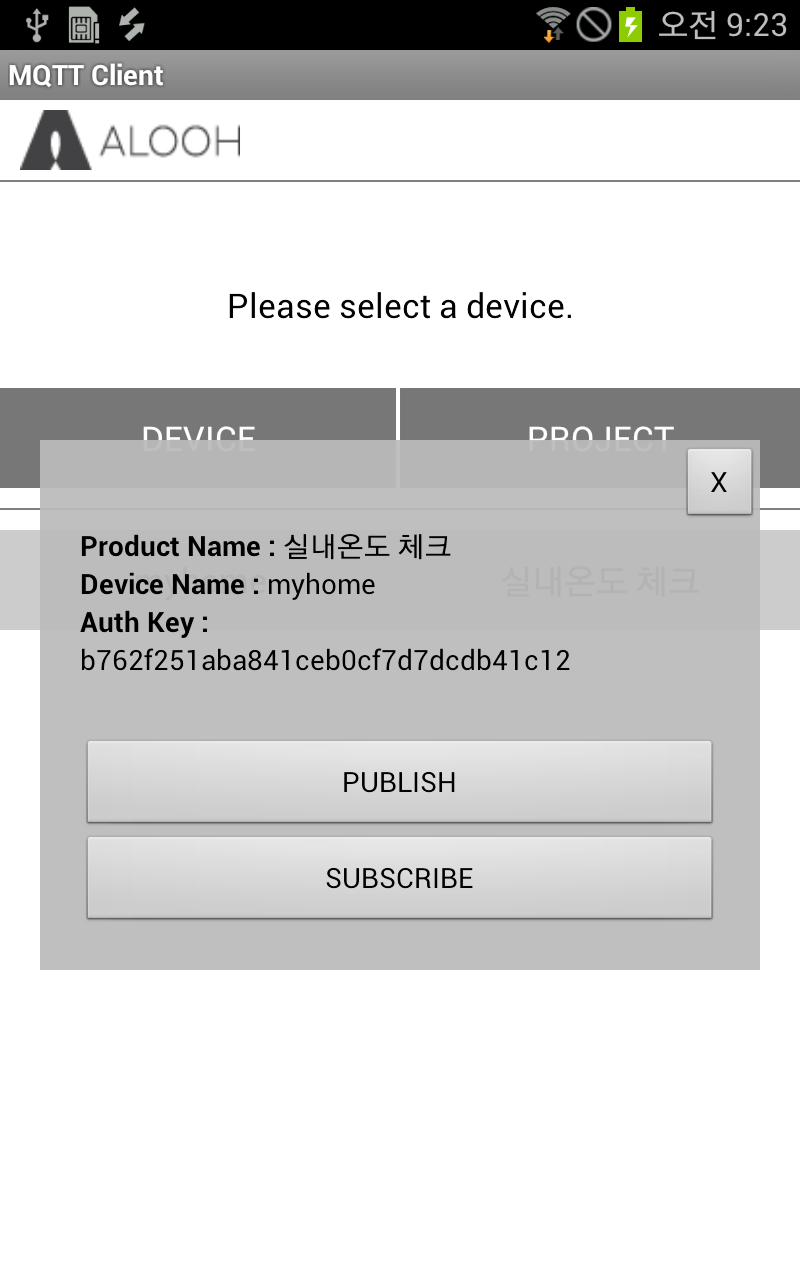
1. Alooh Device 선택

* 안드로이드 Alooh Device 설정 App을 통해 등록된 자신이 보유한 모든 Alooh Device와 속한 프로젝트명과 함께 보여준다. 사용하고자 하는 Alooh Device를 선택을 한다.



1. Publish/Subscribe 선택화면

* 등록된 장치를 선택하면 Publish및 Subscribe 버튼과 더불어 해당 장치에 할당된 인증키값(AuthKey)이 보여진다. 해당 인증키는 Publish나 Subscribe를 수행할 때 Alooh MQTT 서버에서 권한을 확인하기 위한 키로 사용된다.
* PUBLISH : Alooh Cloud 서버를 통해 생성된 프로젝트에는 TOPIC 이라 칭하는 키가 생성이 되는데 특정 TOPIC에 대해 Subscribe를 신청한 Alooh Device에게 메시지를 보내는 역할을 한다. Subscribe를 신청한 여타의 MQTT Client도 메시지 수신이 가능하다.
* SUBSCRIBE : Alooh Device나 여타의 MQTT Client에서 Publish를 통해



1. **DMRController.zip** : 엔진 라이브러리를 이용하는 안드로이드 예제 소스파일

# 이클립스에서의 빌드

## libdmrcp.so

* 이 파일은 안드로이드 프로젝트 내 [libs]/[armeabi] 폴더에 위치해야 한다.
* 아래는 libdmrcp.so 파일을 로드하기 위한 예제 코드이다

static {

System.loadLibrary("dmrcp");

}

## interface.jar

* 이 파일은 안드로이드 프로젝트 내 **[libs]** 폴더에 위치해야 한다.
* 안드로이드 프로젝트 내에 해당 파일을 추가하기 위해 아래 내용을 참조할 수 있다.

1. Project > Properties 선택
2. Java Build Path > Libraries탭 선택
3. Add JARs 버튼 클릭
4. 추가시킬 interface.jar 파일 선택

# APIs

## StartCP

|  |  |  |
| --- | --- | --- |
| Prototype | public static String CPStart(); | |
| **Description** | N-SCREEN CP Engine을 시작한다. | |
| **Parameters** | 없음 | |
| **Return** | String(JSON) | |
| “Result” | Number(1:성공, 0: 실패) |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) |

## StopCP

|  |  |  |
| --- | --- | --- |
| **Prototype** | public static String CPStop() | |
| **Description** | N-SCREEN CP Engine을 중지한다. | |
| **Parameters** | 없음 | |
| **Return** | String(JSON) | |
| “Result” | Number(1:성공, 0: 실패) |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) |

## GetDeviceList

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prototype** | public static String GetDeviceList() | | | |
| **Description** | 현재 로컬네트워크 상에 존재하는 DMR(TV)장치에 대한 정보를 가져온다. | | | |
| **Parameters** | 없음 | | |
| **Return** | String(JSON) | | | |
| “Result” | Number(1:성공, 0: 실패) | | |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) | | |
| “DeviceList”  (array) | “Result” | Number(1: 성공, 0: 실패) | |
| “Name” | String(장치명) | |
| “DeviceId” | String(장치ID) | |
| “Manufacture” | String(제조사) | |
| **Example** | **deviceList = DMRCPInterface.GetDeviceList() ;**  deviceMgr = new DevicesManager();  deviceMgr.getDeivceList().clear();  if(deviceList != null){  try {  JSONObject json = new JSONObject(deviceList);  js\_deviceArray = son.getJSONArray("DeviceList");  for(int i = 0 ; i< js\_deviceArray.length();i++){  device = js\_deviceArray.getJSONObject(i);  m\_pName = device.getString("Name");  m\_pDeviceId = device.getString("DeviceId");  m\_pManufacturer = device.getString("Manufacture");  deviceMgr.getDeivceList().add(deviceMgr.new DeviceInfo(m\_pName, m\_pDeviceId, m\_pManufacture));  }  } catch (JSONException e) { | | | | |

## Play

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | public static String Play(String \_contentsInfo) | | |
| **Description** | 주어진 정보를 기준으로 컨텐츠를 재생한다. Size 값은 String으로 전달된다. | | |
| **Parameters** | String(JSON) | | |
|  | “PlayInfo” | “DeviceId” | String(UDN) |
| “Url” | String(content URL) |
|  | “Title” | String |
|  | “Resolution” | String(“1280X720”) |
|  | “Duration” | String |
|  | “Size” | String |
| **Return** | String(JSON) | | |
| “Result” | | Number(1:성공, 0: 실패) |
| “ErrMessage” | | Number([표1 참조](#_<표_1>_Error)) |
| **Example** | class PlayTask extends AsyncTask<Void, Void, Void>{  @Override  protected Void doInBackground(Void... params) {  JSONObject json = null;  JSONObject playInfo = null;  try {  json = new JSONObject();  json.put("DeviceId",mDeviceMgr.getUdn(mDeviceMgr.getSelectedDevice()));  json.put("Title", mTitle);  json.put("Uri", mUri);  json.put("Duration", mRawDuration);  json.put("Resolution", mResolution);  json.put("Size", mSize);  playInfo = new JSONObject();  playInfo.put("PlayInfo", json);  } catch (JSONException e) {  e.printStackTrace();  }  **DMRCPInterface.Play(playInfo.toString());** | | |

## Stop

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | public static String Stop(String \_deviceId) | | |
| **Description** | 선택된 장치의 재생중인 컨텐츠를 정지시킨다. | | |
| **Parameters** | \_deviceId | 장치의 고유 식별 ID(UDN) |
| **Return** | String(JSON) | | |
| “Result” | Number(1:성공, 0: 실패) | |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) | |

## Pause

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | public static String Pause(String \_deviceId) | | |
| **Description** | 선택된 장치의 재생중인 컨텐츠를 일시정지 시킨다. | | |
| **Parameters** | \_deviceId | 장치의 고유 식별 ID(UDN) |
| **Return** | String(JSON) | | |
| “Result” | Number(1:성공, 0: 실패) | |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) | |

## Resume

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | public static String Resume(String \_deviceId) | | |
| **Description** | 선택된 장치의 일시정지 중인 컨텐츠를 재생 시킨다. | | |
| **Parameters** | \_deviceId | 장치의 고유 식별 ID(UDN) |
| **Return** | String(JSON) | | |
| “Result” | Number(1:성공, 0: 실패) | |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) | |

## Seek

|  |  |  |
| --- | --- | --- |
| **Prototype** | public static String Seek(String \_seekInfo) | |
| **Description** | 재생중인 컨텐츠의 재생위치를 입력된 값으로 변경한다. | |
| **Parameters** | String(JSON) | |
|  | “DeviceId” | String : 장치의 고유식별 ID(UDN) |
| “Position” | String(“hh:mm:ss”) |
| **Return** | String(JSON) | |
| “Result” | Number(1:성공, 0: 실패) |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) |
| **Example** | public void onProgressChanged(SeekBar seekBar, int progress, boolean fromTouch) {  JSONObject jSeek = null;  if(fromTouch == true){  String timestamp = mgr.InttoTimeStamp(progress);  tvProgress.setText(timestamp);  jSeek = new JSONObject();  try {  jSeek.put("DeviceId", mDeviceMgr.getUdn(mDeviceMgr.getSelectedDevice()));  jSeek.put("Position", timestamp);  } catch (JSONException e1) {  e1.printStackTrace();  }  **DMRCPInterface.Seek(jSeek.toString());** | |

## GetVolume

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | public static String GetVolume(String \_deviceId) | | |
| **Description** | 선택된 장치의 볼륨 값을 가져온다. | | |
| **Parameters** | \_deviceId | 장치의 고유 식별 ID(UDN) |
| **Return** | String(JSON) | | |
| “Result” | Number(1:성공, 0: 실패) | |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) | |
|  | “Volume” | String | |

## SetVolume

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | public static String SetVolume(String \_volumeInfo) | | |
| **Description** | 선택된 장치에 입력된 볼륨 값을 설정한다. | | |
| **Parameters** | String(JSON) | |
|  | “DeviceId” | String: 장치의 고유식별 ID(UDN) |
| “Volume” | String |
| **Return** | String(JSON) | | |
| “Result” | Number(1:성공, 0: 실패) | |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) | |

## GetTransportInfo

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | public static String GetTransportInfo(String \_deviceId) | | |
| **Description** | 선택된 장치의 현재 상태를 가져온다. | | |
| **Parameters** | \_deviceId | 장치의 고유 식별 ID(UDN) |
| **Return** | String(JSON) | | |
| “Result” | Number(1:성공, 0: 실패) | |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) | |
| “TransportState” | String("STOPPED","PLAYING","TRANSITIONING",  "PAUSED\_PLAYBACK", "NO\_MEDIA\_PRESENT") | |
| “TransportStatus” | String("OK", "ERROR\_OCCURRED") | |

## GetPositionInfo

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | public static String GetPositionInfo(String \_deviceId) | | |
| **Description** | 선택된 장치의 현재 재생중인 위치 값을 가져온다. | | |
| **Parameters** | \_deviceId | 장치의 고유 식별 ID(UDN) |
| **Return** | String(JSON) | | |
| “Result” | Number(1:성공, 0: 실패) | |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) | |
| “RelTime” | String(“hh:mm:ss”); | |

## GetCurrentMediaInfo

|  |  |  |  |
| --- | --- | --- | --- |
| **Prototype** | public static String GetCurrentMediaInfo(String \_deviceId) | | |
| **Description** | 선택된 장치의 설정된 컨텐츠 정보를 가져온다.  GetTransportInfo()를 통해 재생 여부(“PLAYING”)를 판별 한 후에 호출하는 것을 권장한다. | | |
| **Parameters** | \_deviceId | 장치의 고유 식별 ID(UDN) |
| **Return** | String(JSON) | | |
| “Result” | Number(1:성공, 0: 실패) | |
| “ErrMessage” | Number([표1 참조](#_<표_1>_Error)) | |
| “Title” | String | |
|  | “CurrUrl” | String | |

# CallBack

|  |  |  |
| --- | --- | --- |
| **Description** | 추가 및 삭제된 장치의 정보를 Jar Interface의 CallBack 메소드를 통해서 전달한다.  Application에서는 앱 실행시 CallBack 메소드를 통해서 전달되는 메시지를 수신할 Handler를 등록하여야 한다. | |
| **Message Key** | com.cresprit.cpinterface.DMRCPInterface.DEVICE\_INFO | |
| **Message** | String(JSON) | |
| “Type” | 장치추가 : DMRCPInterface.DMR\_ADD  장치삭제 : DMRCPInterface.DMR\_REMOVE |
| “Name” | String(장치명) |
| “DeviceId” | String(장치ID) |
|  | “Manufacture” | String(제조사) |
| **Example** | **앱실행시**  **DMRCPInterface object = new DMRCPInterface(handler);**  **Handler handler = new Handler(){**  @Override  public void handleMessage(Message msg) {  // TODO Auto-generated method stub  String name = null;  String deviceId = null;  String manufacture = null;  int type=0;//추가 : 0, 삭제: 1  Bundle bundle = msg.getData();  String jsonStr = **bundle.getString(com.cresprit.cpinterface.DMRCPInterface.DEVICE\_INFO);**  **JSONObject json;**  try {  **json = new JSONObject(jsonStr);**  **name = json.getString("name");**  **deviceId = json.getString("deviceId");**  **manufacture = json.getString("manufacture");**  **type = json.getInt("type");**  **if(type == DMRCPInterface.DMR\_ADD)**  deviceMgr.getDeivceList().add(deviceMgr.new DeviceInfo(name, deviceId, manufacture));  Intent intent = new Intent(DEVICE\_MANAGER);  sendBroadcast(intent);  //else Remove DMR Device  // deviceMgr.getDeviceList().del  } catch (JSONException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  super.handleMessage(msg);  }}; | |

# 구현 가이드

## DMR CP Engine 실행순서

1. dmrcp 라이브러리 로드
2. Starting Engine

(3) 장치리스트 획득

(4) 재생 대상장치(TV) 선택 후 Play 시도

(5) GetPositionInfo를 2~3초 단위로 반복 호출, 대상장치에서의 현재 재생위치 값을

추출하여 안드로이드 App의 프로그레스바에 반영

## Example Source Code

//Start Engine

class EngineStartTask extends AsyncTask<Void, Void, String>{

JSONObject json= null;

String jReturnValue;

String result;

String errMsg;

@Override

protected String doInBackground(Void... params) {

// TODO Auto-generated method stub

jReturnValue = DMRCPInterface.*StartCP();*

try {

json = new JSONObject(jReturnValue);

result = json.getString("Result");

errMsg = json.getString("ErrMessage");

…………………………………………………………………………………………………..

//Get DeviceList

deviceList = DMRCPInterface.*GetDeviceList*() ;

deviceMgr = **new** DevicesManager();

deviceMgr.getDeivceList().clear();

**if**(deviceList != **null**)

{

**try** {

JSONObject json = **new** JSONObject(deviceList);

js\_deviceArray = json.getJSONArray("DeviceList");//.getJSONObject("DeviceList");

**for**(**int** i = 0 ; i< js\_deviceArray.length();i++)

{

device = js\_deviceArray.getJSONObject(i);

pName = device.getString("name");

pDeviceId = device.getString("deviceId");

deviceMgr.getDeivceList().add(deviceMgr.**new** DeviceInfo(pName, pDeviceId));

}

…………………………………………………………………………………………………..

//Playing

**class** PlayTask **extends** AsyncTask<Void, Void, Void>{

@Override

**protected** Void doInBackground(Void... params) {

// **TODO** Auto-generated method stub

JSONObject json = **null**;

JSONObject playInfo = **null**;

**try** {

json = **new** JSONObject();

json.put("deviceId", mDeviceMgr.getUdn(mDeviceMgr.getSelectedDevice()));

json.put("uri", mUri);

json.put("duration", mRawDuration);

json.put("size", mSize);

playInfo = **new** JSONObject();

playInfo.put("PlayInfo", json);

} **catch** (JSONException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

DMRCPInterface.*Play*(playInfo.toString());

…………………………………………………………………………………………………..

taskHandler.postDelayed(startIntervalRunnable, 3000);

task = new PlayTask();

private final Runnable startIntervalRunnable = new Runnable() {

public void run() {

String transportInfo =null;

transportInfo = DMRCPInterface.*GetTransportInfo*(mDeviceMgr.getUdn(mDeviceMgr.getSelectedDevice()));

……………………..

if(!"".equals(mPlayingStatus) && mPlayingStatus != null)

{

if(mPlayingStatus.equals("PLAYING"))//재생중이면 반복적으로 현재 재생위치 확인

taskHandler.postDelayed(mCheckPlayingRunnable, 500);

else

taskHandler.postDelayed(startIntervalRunnable, 3000);

}

else

{

taskHandler.postDelayed(startIntervalRunnable, 3000);

}

……………………..

private final Runnable mCheckPlayingRunnable = new Runnable() {

public void run() {

. . . . . . . . .

transportInfo = DMRCPInterface.*GetTransportInfo*(mDeviceMgr.getUdn(mDeviceMgr.getSelectedDevice()));

if(transportInfo != null){

try {

JSONObject json = new JSONObject(transportInfo);

mPlayingStatus = json.getString("TransportState");

} catch (JSONException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

if(!"".equals(mPlayingStatus) && mPlayingStatus != null){

if(mPlayingStatus.equals("PLAYING")){

positionInfo = DMRCPInterface.*GetPositionInfo*(mDeviceMgr.getUdn(mDeviceMgr.getSelectedDevice()));

if(positionInfo != null){

try {

JSONObject json = new JSONObject(positionInfo);

position = json.getString("RelTime");

} catch (JSONException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

if(!"".equals(position) && position != null)

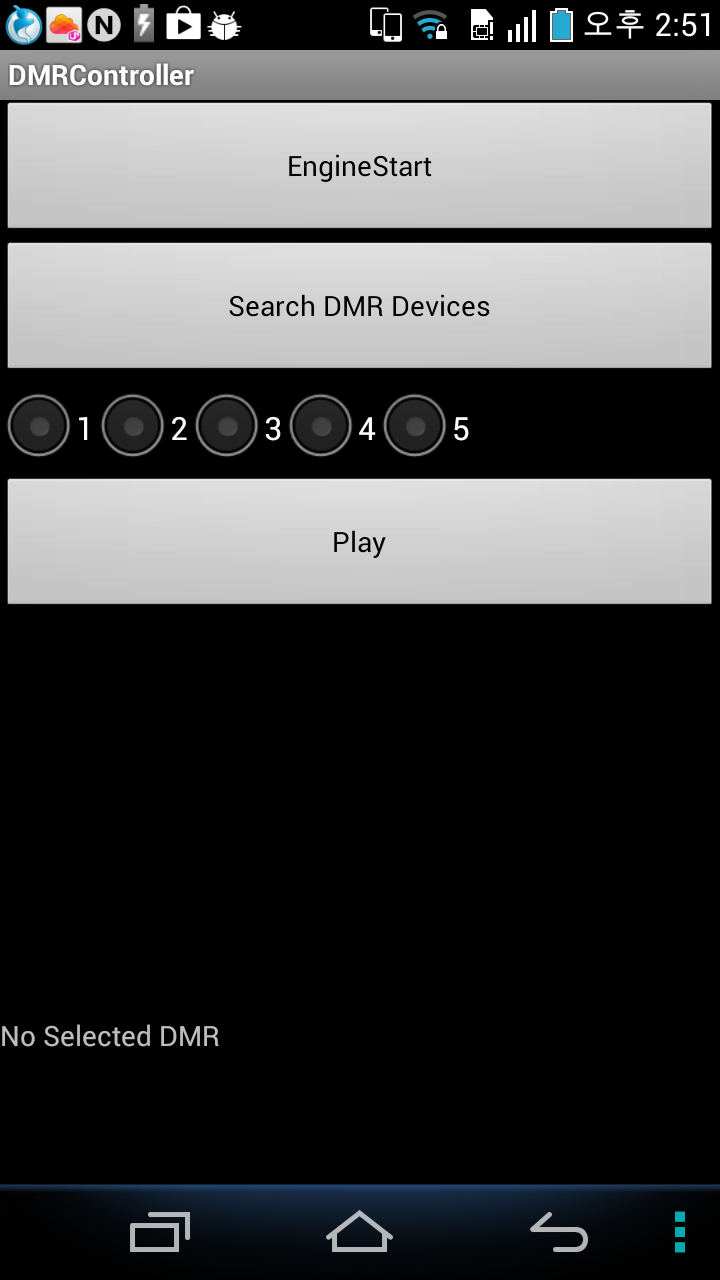
{

curPosition = mgr.TimeStamptoInt(position);

mSeekBar.setProgress(curPosition);

……………………..

# Sample Application Guide



<Sample Application 시작화면>

## EngineStart

: N-Screen 엔진을 구동시킨다. 동일 네트워크상에 있는 다른 DMR(TV)장치로부터 alive 메시지를 받을 수 있다.

## Search DMR Devices

: N-Screen 엔진이 정상적으로 구동이 된 후, 검색된 장치들을 DMRCPInterface의 CallBack 메소드를 통하여 등록된 Application의 Handler로 전달한다. Application에서는 검색된 장치들을 화면에 표시한다.

## Radio Button

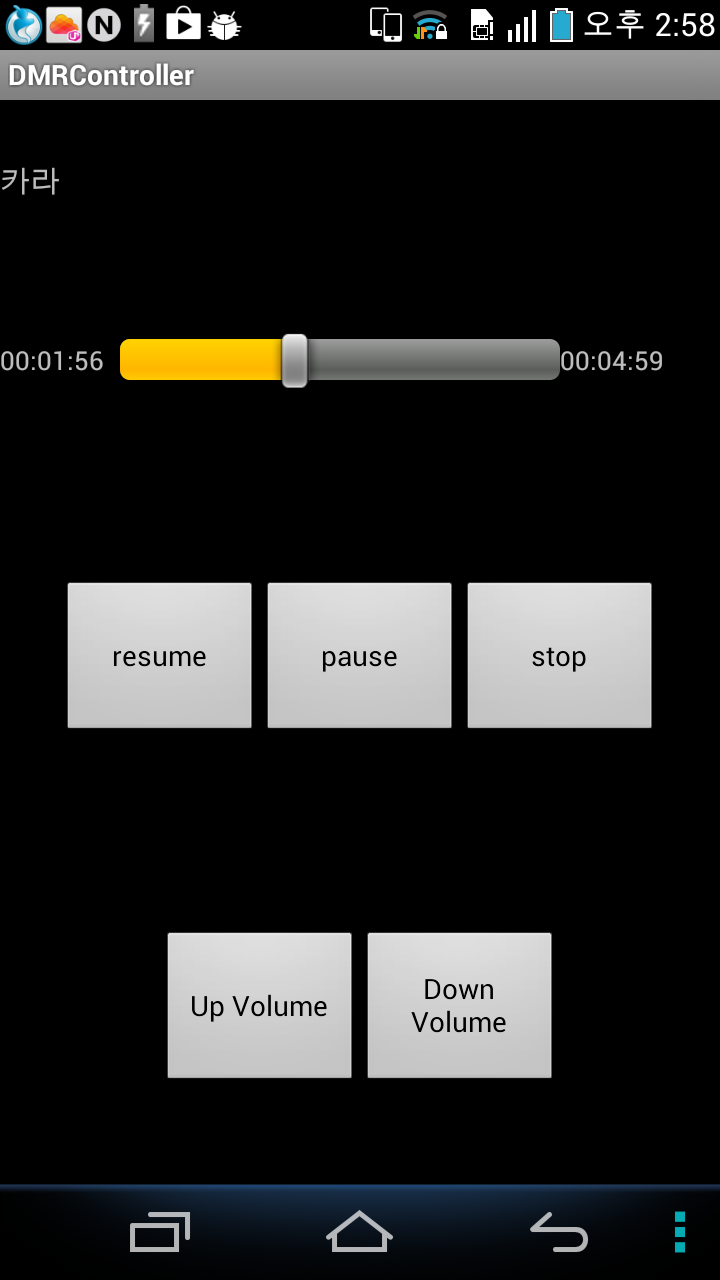
: 각각의 RadioButton은 특정 컨텐츠를 나타낸다.

## Play

: DMR(TV)장치를 선택한 후 컨텐츠에 대한 정보를 바탕으로 재생을 시도한다. 시도와 동시에 플레이어 화면으로 진입한다.

## No Selected DMR

: 6.2를 통해 장치를 선택하면 선택된 장치명이 표시된다.



<컨텐츠 재생중인 플레이어 화면>

* 프로그레스바의 진행은 반복적으로 GetPositionInfo() API를 통해 DMR(TV)장치에서의 현재 위치값을 얻은 후 적용함으로서 처리된다.
* 프로그레스바를 터치해서 Seek() 동작을 할 수 있다.
* Pause : 일시정지
* Resume : 다시재생
* Stop : 재생중지
* UP/Down Volume : 볼륨 조절

# <표 1> Error Message

#define ENGINE\_E\_SUCCESS 0

#define ENGINE\_E\_NULL\_PARAM -1

#define ENGINE\_E\_NOT\_RUNNING -2

#define ENGINE\_E\_INVALID\_HANDLE -100

#define ENGINE\_E\_INVALID\_PARAM -101

#define ENGINE\_E\_OUTOF\_HANDLE -102

#define ENGINE\_E\_OUTOF\_CONTEXT -103

#define ENGINE\_E\_OUTOF\_MEMORY -104

#define ENGINE\_E\_INIT -105

#define ENGINE\_E\_BUFFER\_TOO\_SMALL -106

#define ENGINE\_E\_INVALID\_DESC -107

#define ENGINE\_E\_INVALID\_URL -108

#define ENGINE\_E\_INVALID\_SID -109

#define ENGINE\_E\_INVALID\_DEVICE -110

#define ENGINE\_E\_INVALID\_SERVICE -111

#define ENGINE\_E\_BAD\_RESPONSE -113

#define ENGINE\_E\_BAD\_REQUEST -114

#define ENGINE\_E\_INVALID\_ACTION -115

#define ENGINE\_E\_FINISH -116

#define ENGINE\_E\_INIT\_FAILED -117

#define ENGINE\_E\_URL\_TOO\_BIG -118

#define ENGINE\_E\_BAD\_HTTPMSG -119

#define ENGINE\_E\_ALREADY\_REGISTERED -120

#define ENGINE\_E\_INVALID\_INTERFACE -121

#define ENGINE\_E\_NETWORK\_ERROR -200

#define ENGINE\_E\_SOCKET\_WRITE -201

#define ENGINE\_E\_SOCKET\_READ -202

#define ENGINE\_E\_SOCKET\_BIND -203

#define ENGINE\_E\_SOCKET\_CONNECT -204

#define ENGINE\_E\_OUTOF\_SOCKET -205

#define ENGINE\_E\_LISTEN -206

#define ENGINE\_E\_TIMEDOUT -207

#define ENGINE\_E\_SOCKET\_ERROR -208

#define ENGINE\_E\_FILE\_WRITE\_ERROR -209

#define ENGINE\_E\_CANCELED -210

#define ENGINE\_E\_EVENT\_PROTOCOL -300

#define ENGINE\_E\_SUBSCRIBE\_UNACCEPTED -301

#define ENGINE\_E\_UNSUBSCRIBE\_UNACCEPTED -302

#define ENGINE\_E\_NOTIFY\_UNACCEPTED -303

#define ENGINE\_SOAP\_E\_INVALID\_ACTION -401

#define ENGINE\_SOAP\_E\_INVALID\_ARGS -402

#define ENGINE\_SOAP\_E\_OUT\_OF\_SYNC -403

#define ENGINE\_SOAP\_E\_INVALID\_VAR -404

#define ENGINE\_E\_INVALID\_ARGUMENT -501

#define ENGINE\_E\_FILE\_NOT\_FOUND -502

#define ENGINE\_E\_FILE\_READ\_ERROR -503

#define ENGINE\_E\_EXT\_NOT\_XML -504

#define ENGINE\_E\_NO\_WEB\_SERVER -505

#define ENGINE\_E\_OUTOF\_BOUNDS -506

#define ENGINE\_E\_NOT\_FOUND -507

#define ENGINE\_SOAP\_E\_ACTION\_FAILED -509

#define ENGINE\_E\_INTERNAL\_ERROR -911