**Roundee.io Android API 메뉴얼**

**Version 1.6**

**문서 요약**

본 문서는 Roundee.io platform에서 제공하는 Android API에 설명서이다.

**문서 이력**

|  |  |  |  |
| --- | --- | --- | --- |
| **버전** | **날짜** | **작성자** | **변경 내역** |
| 1.0 | 2018. 8. 22 | 이상혁 | 최초 작성 |
| 1.1 | 2019. 2. 19 | 이상혁 | 기능추가(레코딩,북마크,채팅) |
| 1.2 | 2019. 5. 09 | 이상혁 | Android Studio 설정추가 |
| 1.3 | 2019. 5. 23 | 이상혁 | Error Callback 추가 CallLog 기능 추가 Speak Mode변경 추가 Custom RoomID make 추가 Android 9.0 이슈 수정 (안드로이드 설정 websocket 참고) |
| 1.4 | 2019.6.10 | 이상혁 | Startmedia 해상도 관련  parameter 삭제 |
| 1.5 | 2019.7.12 | 이상혁 | 해상도 , Bitrate , Frame  변경기능 추가 |
| 1.6 | 2019.7.30 | 이상혁 | Log 파일 생성기능 추가  Log on/off 기능 추가 |

**목 차**

[**1.** **개요** 6](#_Toc15397947)

[1.1. 기본 API 동작 Flow 6](#_Toc15397948)

[1.2. 기본 서비스 절차 6](#_Toc15397949)

[**2.** **개발환경 설정** 9](#_Toc15397950)

[2.1. 이클립스 설정방법 9](#_Toc15397951)

[2.2. Android Studio 설정방법 10](#_Toc15397952)

[**3.** **주요 클래스** 20](#_Toc15397953)

[3.1. Roundee Class 20](#_Toc15397954)

[3.2. SttMyInfo 21](#_Toc15397955)

[3.3. SttRoonInfo Class 21](#_Toc15397956)

[3.4. SttMember Class 22](#_Toc15397957)

[3.5. SttMemberList Class 22](#_Toc15397958)

[3.6. Notification Class 22](#_Toc15397959)

[3.7. SttUploadResult Class 22](#_Toc15397960)

[3.8. SttRecord Class 23](#_Toc15397961)

[3.9. SttRecordList Class 23](#_Toc15397962)

[3.10. SttChatMsg Class 23](#_Toc15397963)

[3.11. SttChatMsgList Class 24](#_Toc15397964)

[3.12. SttCallLog Class 24](#_Toc15397965)

[**4.** **주요 콜백** 25](#_Toc15397966)

[4.1. RoundeeCallBack.WebSocketResult 25](#_Toc15397967)

[4.2. RoundeeCallBack.CallbackResult 25](#_Toc15397968)

[4.3. RoundeeCallBack.RoomCallbackResponse 25](#_Toc15397969)

[4.4. RoundeeCallBack.NotiResponse 25](#_Toc15397970)

[4.5. RoundeeCallBack.FileUploadCallback 27](#_Toc15397971)

[**5.** **클래스 상세설명** 28](#_Toc15397972)

[5.1. Function 28](#_Toc15397973)

[5.1.1. getInstance 28](#_Toc15397974)

[5.1.2. connectToServer 28](#_Toc15397975)

[5.1.3. Login 29](#_Toc15397976)

[5.1.4. regNotiListener 29](#_Toc15397977)

[5.1.5. CreateRoom 30](#_Toc15397978)

[5.1.6. getRoomInfo 31](#_Toc15397979)

[5.1.7. StartMedia 32](#_Toc15397980)

[5.1.8. JoinRoom 33](#_Toc15397981)

[5.1.9. AddViewer 33](#_Toc15397982)

[5.1.10. changeUserViewer 34](#_Toc15397983)

[5.1.11. updateDisplayName 35](#_Toc15397984)

[5.1.12. switchCamera 35](#_Toc15397985)

[5.1.13. getScreenLayout 36](#_Toc15397986)

[5.1.14. setMediaMute 36](#_Toc15397987)

[5.1.15. removeViewer 37](#_Toc15397988)

[5.1.16. InviteUser 37](#_Toc15397989)

[5.1.17. LeaveRoom 38](#_Toc15397990)

[5.1.18. exitConference 38](#_Toc15397991)

[5.1.19. logout 39](#_Toc15397992)

[5.1.20. SendChatMsg 39](#_Toc15397993)

[5.1.21. recordingVideo 40](#_Toc15397994)

[5.1.22. addBookmarkMsg 40](#_Toc15397995)

[5.1.23. upLoadFile 41](#_Toc15397996)

[5.1.24. downLoadFile 41](#_Toc15397997)

[5.1.25. getCallLog 42](#_Toc15397998)

[5.1.26. setSpeakerMode 42](#_Toc15397999)

[5.1.27. setMuteUser 43](#_Toc15398000)

[5.1.28. setDetailVideoQuality 44](#_Toc15398001)

[5.1.29. setSimpleVideoQuality 45](#_Toc15398002)

[5.1.30. setDebuggable 46](#_Toc15398003)

[5.1.31. setDebugMakeFile 46](#_Toc15398004)

[**6.** **Error Code 명세** 47](#_Toc15398005)

# **개요**

Roundee.io Android API는 WebRTC를 이용하여 1:n Video Conference(SFU) 및 Conference 정보를 Get/Set 위한 Signal 기능을 제공하기 위해 제작된 라이브러리이다.

본 API는 Android(java)로 개발되어 있으며, Android 4.4 버전 이상에서 동작한다..

* SFU : Selective Forwarding Unit

## 기본 API 동작 Flow

기본적으로 애플리케이션에서 필요한 기능을 수행하기 위해 각종 Function 들을 호출하며, 결과값들은 Callback을 통해 구현한다.

Roundee.io 서비스 제어를 위해서는 Signal Server와 연동하며, 미디어 서비스 제어를 위해서는 Media Server와 연동한다.



## 기본 서비스 절차

Roundee.io 서비스는 서비스 이용자를 Host와 Guest로 구분한다. Host는 Conference room을 개설하는 이용자를 의미하며, Guest는 Host가 개설한 Conference room에 참여(Join)하는 이용자를 의미한다.

Roundee.io에서 제공하는 기본 서비스 절차는 아래 그림과 같다. Host는 Video Conference를 생성하는 사용자이며, Guest를 Video Conference에 참여하는 사용자를 의미한다.

* Video Conference 생성

Host는 Video Conference를 생성하기 위해, Signal Server에 접속 및 인증 과정을 거쳐야 한다.

생성된 Video Conference room에 Guest를 초대하기 위한 InviteUser() Function은 Guest에게 이메일을 보내는 형식으로 되어 있으며, 이메일이 아닌 다른 방식을 사용하는 경우에는 임의의 다른 방식의 사용이 가능하다.



* Video Conference 참여

Guest는 생성된 Video Conference에 참여하기 위해, Signal Server에 접속 및 인증 과정을 거쳐야 한다. Guest는 전달받은 url 정보를 이용하여 Video Conference에 참여한다.

Video Conference는 Guest의 참여 수에 따라 1:1, 1:N으로 구분된다.

[Video Conference 참여 (1:1)]



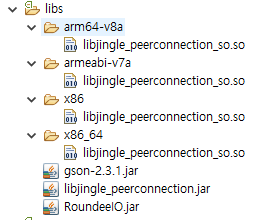
[Video Conference 참여 (1:N) – Multi View]



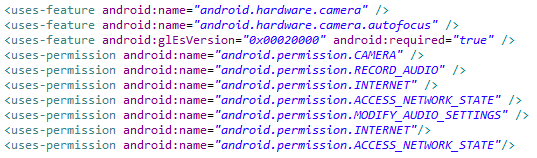
# **개발환경 설정**

## 이클립스 설정방법

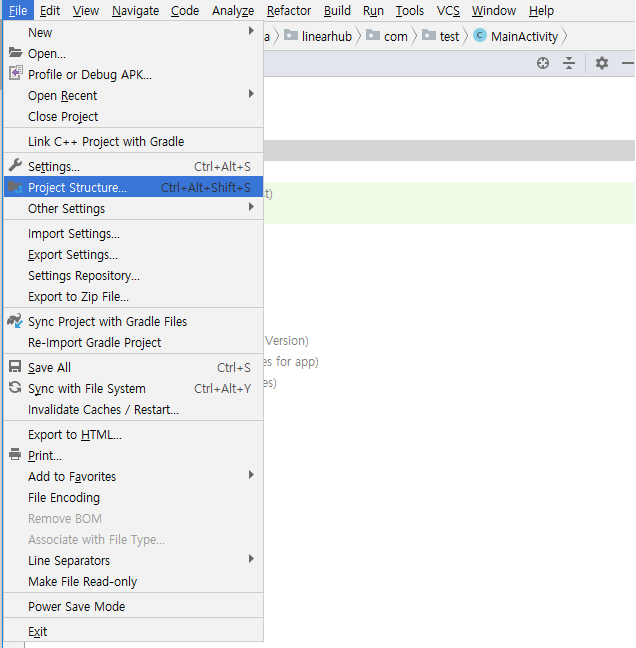
Roundee.io API를 이용하여 Android App.을 개발하기 위해서는, 제공된 라이브러리를 다음과 같이 추가하여야 한다.



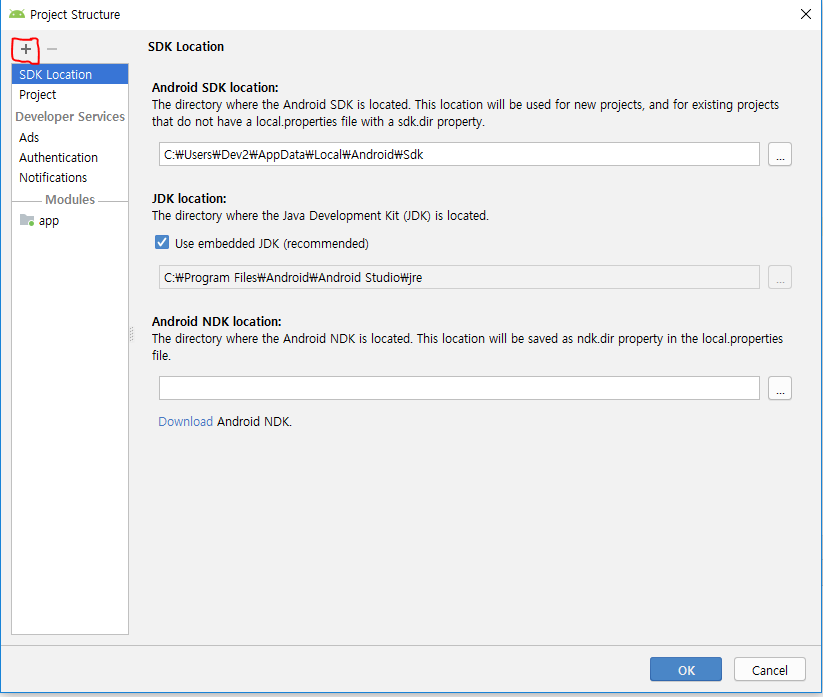
Camera , Mic , Internet 사용을 위해 다음과 같이 Permission 을 추가한다.



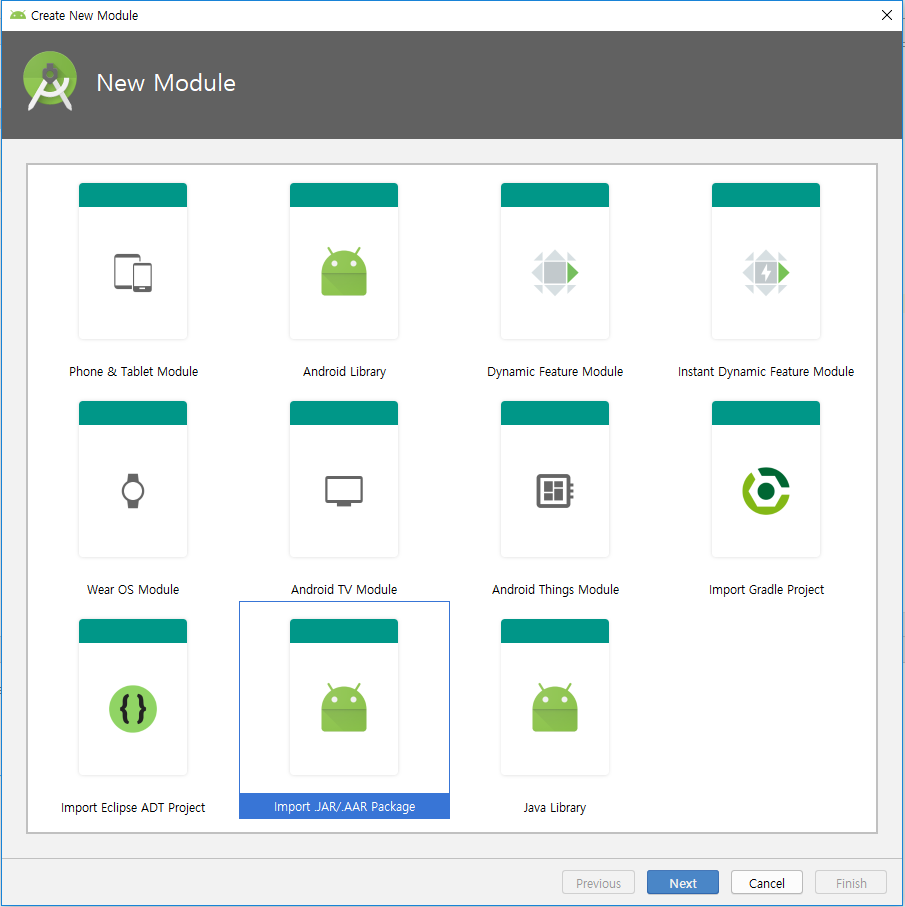
## Android Studio 설정방법



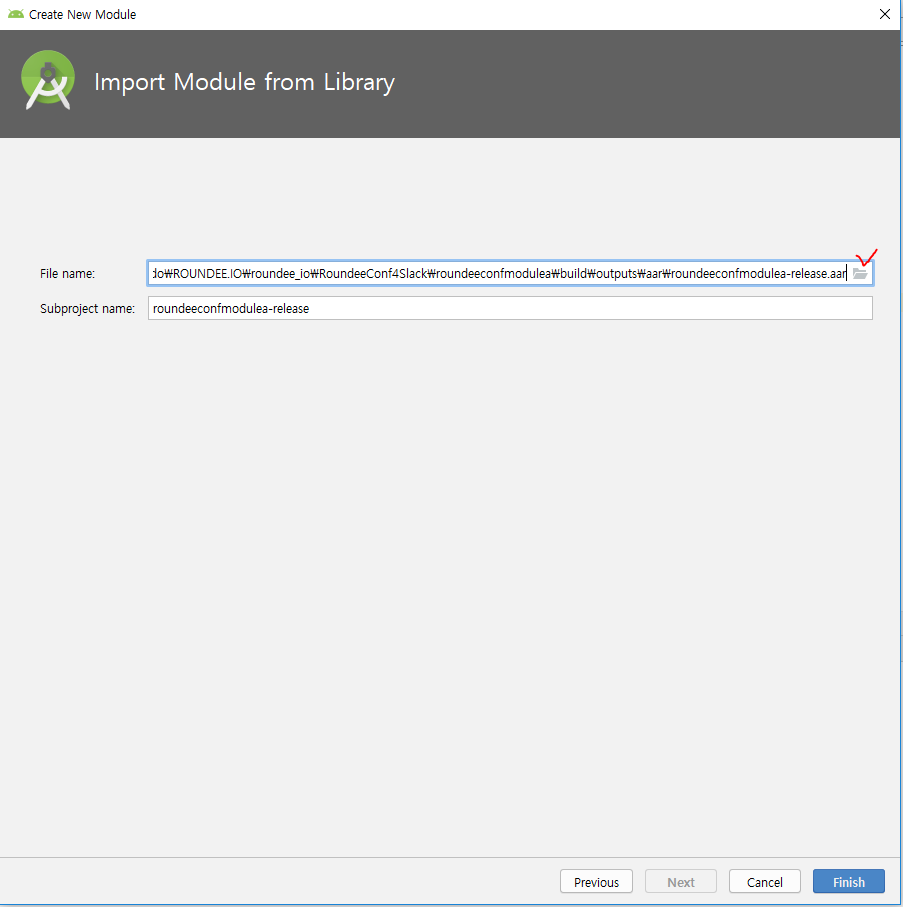
* 새로운 프로젝트 생성 후 File 메뉴에서 Project Structure… 선택

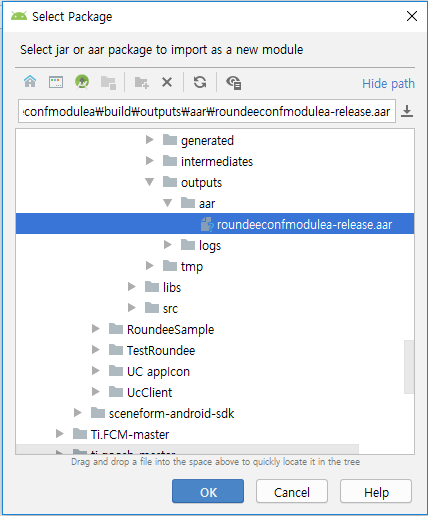


* 왼쪽 상단에 ‘+’ 버튼을 클릭 후 모듈을 추가

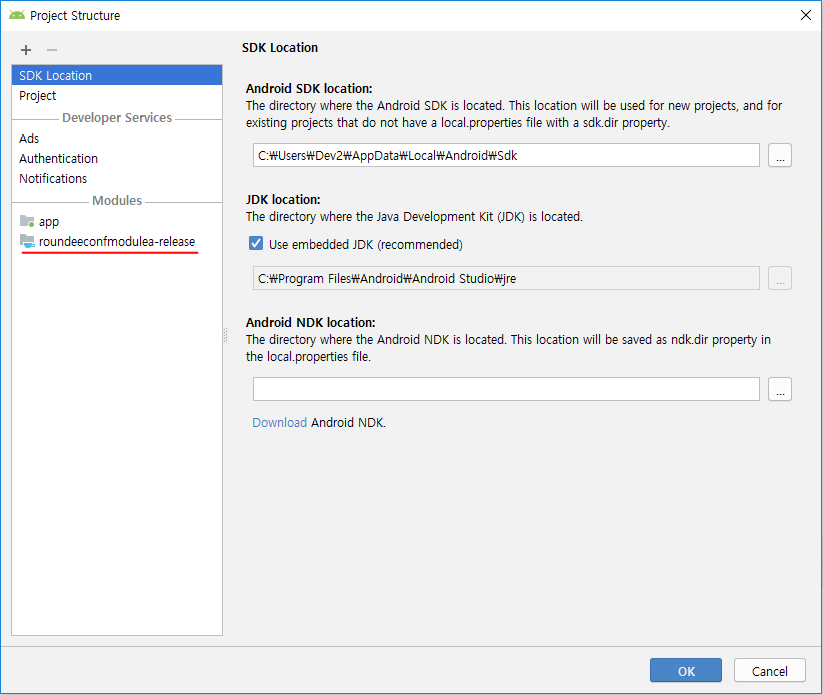


* Import JAR/AAR Package 선택

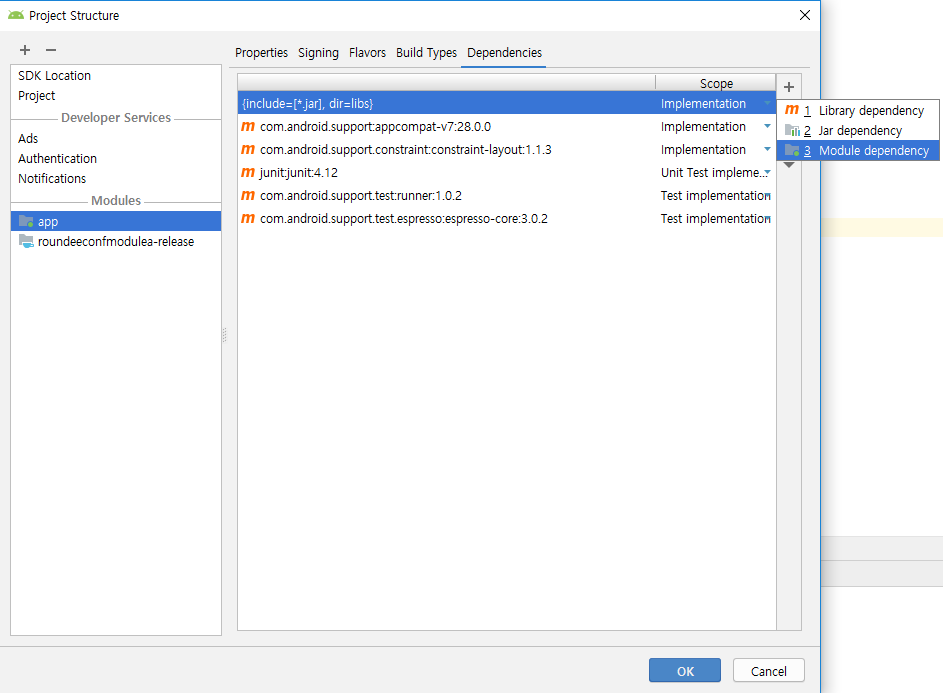




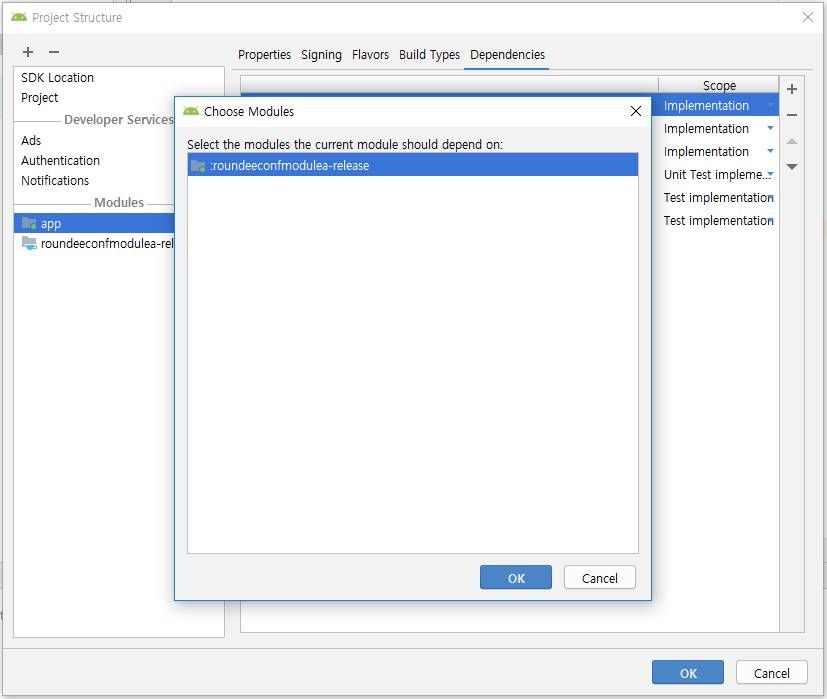
* 전달해준 roundeeconfmodulea-release.arr 파일 선택



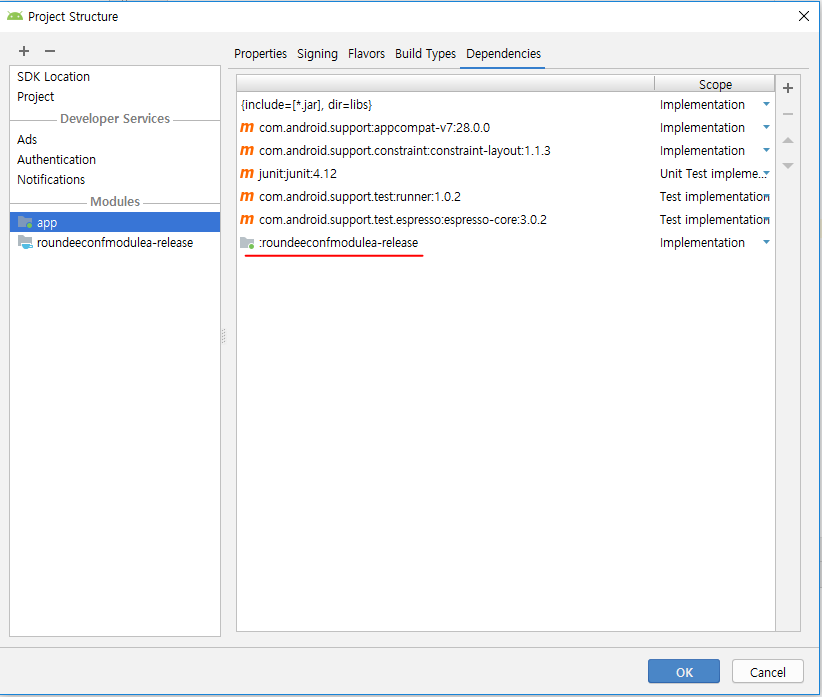
* Modules에 roundeeconfmodulea-release 추가되었는지 확인



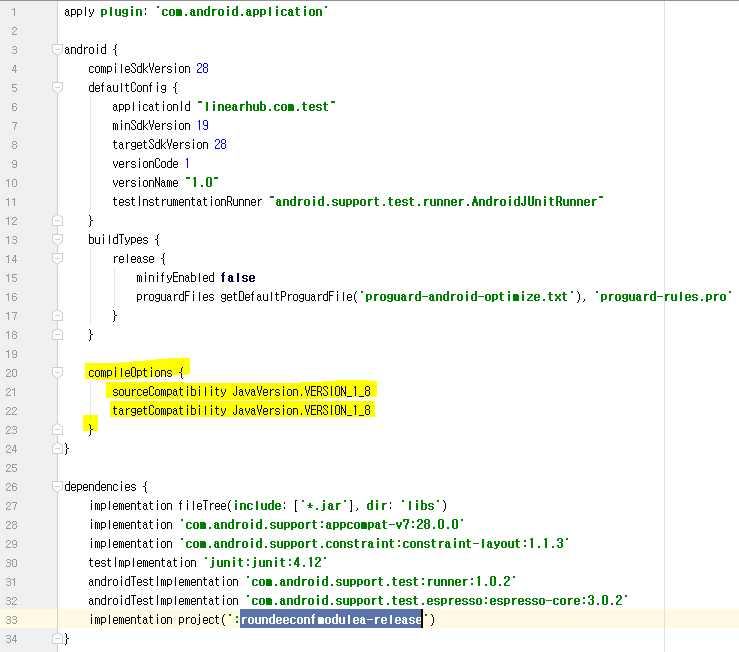
* Modules 에서 app 선택 후 우측상단의 Dependencies 탭에서 ‘+’ 버튼 클릭 후 Module Dependency 선택



* 추가한 모듈 선택



* Dependencies 에 roundeeconfmodulea-release 추가 되었는지 확인



* app build.gradle 파일 수정(compileOptions 추가)

compileOptions {

sourceCompatibility JavaVersion.VERSION\_1\_8

targetCompatibility JavaVersion.VERSION\_1\_8

}



* Android 9.0 에서 WebSocket 연결

<uses-library android:name="org.apache.http.legacy" android:required="false" /> 추가

# **주요 클래스**

## Roundee Class

Roundee Class는 서비스 동작을 위한 모든 멤버 함수들을 포함하고 있는 메인 클래스이다.

Singleton으로 되어 있으며, getInstance(Context con)로 호출하여 사용한다. (예 : Roundee.getInstance(this))

Roundee 클래스의 멤버 함수들은 다음과 같다.

|  |  |
| --- | --- |
| **클래스** | **설명** |
| getInstance | Roundee 생성자 |
| connectToServer | Signal Server 주소 설정 및 연결 |
| Login | Signal Server 로그인 |
| regNotiListener | Conference Room 관련 Noti수신 Listener 등록 |
| CreateRoom | Conference Room 생성 |
| getRoomInfo | Conference Room Info 조회 |
| StartMedia | 디바이스 Camera open 및 서버와 미디어 연결 |
| JoinRoom | Conference Room 입장 |
| AddViewer | Conference 참여자중 원하는 user 화면에 추가 |
| changeUserViewer | 이미 보여지고 있는 화면을 다른user로 변경 |
| updateDisplayName | 상대방에 표시될 displayName 변경 |
| switchCamera | 카메라 Front / rear 변경 |
| getScreenLayout | 해당 user가 보여지고 있는 layout 조회 |
| setMediaMute | 미디어 on/off |
| removeViewer | 해당 뷰의 Connection 종료 |
| InviteUser | Conference Room 으로 user초대 |
| LeaveRoom | Conference Room 나가기 |
| exitConference | Conference Room 종료 |
| logout | Signal Server 로그아웃 |
| sendChatMsg | Chat 메시지 전달 |
| recordingVideo | Conference Room 녹화시작 / 녹화중지 |
| addBookmarkMsg | Conference Room 영상 녹화중 북마크 추가 |
| upLoadFile | 파일 업로드 |
| downLoadFile | 파일 다운로드 |
| getCallLog | 통화 내역 확인 |
| setSpeakerMode | 스피커폰 모드 On/Off |

## SttMyInfo

SttMyInfo Class는 사용자 데이터를 가지고 있는 Object Class이다.

* + public String comId : comId
  + public String dep : depId
  + public String utcUs : TimeZone 정보
  + public String token : 사용자 토큰
  + public String myid : 유저 Id

## SttRoonInfo Class

SttRoonInfo Class는 Conferenceroom 관련 데이터들을 가지고 있는 Object Class이다.

* + public String desc : 방 설명
  + public String end\_time : 종료시간
  + public String name : 방이름
  + public String owner : 방 생성한 유저ID
  + public String room\_id : 방ID
  + public String start\_time : 시작시간
  + public String state : Conference 상태 (schedule - 예약, progress - 생성, end - 종료)
  + public String time\_offset : 타임 offset(한국 +9) 설정
  + public String time\_zone : 타임존
  + public ArrayList<SttMember> cmembers : Sttmember List

## SttMember Class

SttMember Class는 유저관련 데이터들을 가지고 있는 Object Class이다.

* + public String userid : 유저ID
  + public String state : 유저상태(ex join / exit)
  + public int type = 0 : 유저타입(0:general , 1:viewer)
  + public String displayname : 상대방에 보여줄 이름
  + public String quality : 카메라 화질(ex high , low)
  + public String mstate : 미디어 상태(ex audioonly , videoonly , none)

## SttMemberList Class

SttMemberList Class는 SttMember 리스트를 가지고 있는 Object Class이다.

* + ArrayList<SttMember> mSttMemberList

## Notification Class

Notification Class는 Notification 종류를 가지고 있는 enum class이다.

* + USER\_STATE (유저 상태가 변경되었음을 알림)
  + USER\_INFO (유저 정보가 변경되었음을 알림 ex. displayname 변경)
  + USER\_MEDIA\_STATE (유저 미디어 상태가 변경되었음을 알림)
  + ROOM\_STATE (방 상태가 변경되었음을 알림)
  + CHAT\_MSG(채팅 메시지 수신을 알림)
  + RECORDING(레코딩 상태변경 알림)

## SttUploadResult Class

SttUploadResult Class는 파일 업로드관련 데이터를 가지고있는 Object Class이다.

* + public boolean success : 성공/실패
  + public String reason : 실패 이유
  + public String mOriginDownLoadPath : 원본파일 주소
  + public String mThumnailDownLoadPath : 섬네일 주소
  + public int mFileSize : 파일 사이즈
  + public String mOriginFileName : 원본 파일 이름
  + public String mExtention : 파일 확장자
  + public String mType : 파일 타입(이미지/기타)
  + public String mMsgType : 메시지 타입(image / etc)
  + public boolean hasThum : 섬네일 유무

## SttRecord Class

SttRecord Class는 미디어 녹화 관련 데이터를 가지고 있는 Object Class이다.

* + public String userid : 유저ID
  + public String start\_time : 녹화 시작 시간
  + public String end\_time : 녹화 종료 시간

## SttRecordList Class

SttRecordList Class는 SttRecord 리스트를 가지고 있는 Object Class이다.

* + public String recording : 녹화중
  + public ArrayList<SttRecord> recordings

## SttChatMsg Class

SttChatMsg Class는 채팅 메시지 전송할 데이터를 가지고 있는 Object Class이다.

* + public String roomID : 방ID
  + public String msgID : 메시지ID
  + public String senderName : 보내는 사람이름
  + public String sender : 보내는 사람ID
  + public String msgDateTime : 메시지 보낸 시간
  + public String msgData : 메시지 내용
  + public String fileExtenstion : 파일 확장자
  + public String msgType : 메시지 타입(text , note , image , doc)
  + public String type : 메시지 타입(text , note , image , doc)
  + public String originalUrl : 원본파일 URL
  + public String thumbnailUrl : 섬네일파일 URL
  + public String recordtimer : 북마크시간 (초단위 , ex : 30)

## SttChatMsgList Class

SttChatMsgList Class는 SttChatMsg 리스트를 가지고 있는 Object Class이다.

* + public ArrayList<SttChatMsg> sttChatMsgArrayList

## SttCallLog Class

SttCallLog Class 는 통화관련 데이터를 가지고 있는 Object Class 이다.

* + public String Call\_type : CallType (수신 : incoming , 발신 : outgoing)
  + public long Create\_Room\_time : 방생성 시간(UCT , 발신일 경우에만 값이 존재함)
  + public long Join\_Room\_Time : 방입장 시간(UCT)
  + public long Peer\_Join\_Room\_Time : 상대방 방 입장시간(UCT , 발신일 경우에만 값이 존재함)
  + public long Exit\_Call\_Time : 통화 종료시간(UCT , 통화를 먼저 끊는 사람에게만 존재함)
  + public long Peer\_Exit\_Conference : 상대방 통화 종료 시간(UCT , 상대방이 먼저 통화를 끊을 경우에만 존재함)
  + public Call\_Duration : 통화시간(ms)
  + public Boolean useLog : 로그사용여부(라이브러리 자체적으로만 사용됨)

# **주요 콜백**

Roundee 사용시 결과 값을 리턴 받는 Callback들에 대한 설명이다.

## RoundeeCallBack.WebSocketResult

* Signal Server와 연결을 위해 사용하는 Web Socket CallBack을 받음.

|  |  |
| --- | --- |
| **Function** | **설명** |
| onConnect() | Signal Server 연결 성공시 호출 |
| onDisConnect(int Code, String reason) | Signal Server 연결이 끊어졌을 때 호출 |
| onError(int code, String reason) | Signal Server 접속실패시 호출 |

## RoundeeCallBack.CallbackResult

* API Function Request의 결과를 CallBack으로 받음.

|  |  |
| --- | --- |
| **Function** | **설명** |
| onResult(boolean success, int errorcode, String reason) | 성공/실패의 결과값이 호출된다.  성공인 경우, errorcode 값은 0 |

## RoundeeCallBack.RoomCallbackResponse

* Conference 정보 요청에 대한 CallBack을 받음.

|  |  |
| --- | --- |
| **Function** | **설명** |
| onResponse(SttRoomInfo sttRoomInfo, SttMemberList sttMemberList) | Conference room 데이터 호출 |

## RoundeeCallBack.NotiResponse

* Signal Server로부터 내려 오는 Notification을 받음.

|  |  |
| --- | --- |
| **Function** | **설명** |
| getMessage(String roomId, Noti noti , SttRoomInfo sttRoomInfo, Sttmember sttMember , SttChatMsg sttChatMsg , SttRecordList sttRecordList , SttConvertFile mSttConvertFile , SttConverterData mSttConverterData , SttKnock mSttKnock) | Noti 메시지 수신시 호출 |

## RoundeeCallBack.FileUploadCallback

* 파일 업로드 요청에 대한 CallBack을 받음

|  |  |
| --- | --- |
| **Function** | **설명** |
| onProgress(int progrerss) | 파일 업로드 진행상태 업데이트 |
| onResult(SttUploadResult s) | 파일 업로드 결과값 수신시 호출 |

# **클래스 상세설명**

Roundee.io를 사용하기 위한 Roundee 클래스의 멤버함수에 대한 설명이다.

## Function

### getInstance

Roundee Signleton 생성자이며, Roundee 멤버함수 들을 사용할 때 항상 사용하여야 한다.

* 예제

|  |
| --- |
| Roundee.getInstance(this); |

### connectToServer

Signal Server를 연결하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| serverAddr | String | Signal Server 주소 |
| port | Int | Signal Server 포트 |
| RoundeeCallback.WebSocketResult | callback | 웹소켓 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).connectToServer(**“dev0.roundee.com”**, 7060, **new** RoundeeCallback.WebSocketResult() {  @Override  **public void** onConnect() {  }  @Override  **public void** onDisconnect(**int** code, String reason) {  }  @Override  **public void** onError(**int** code, String reason) {  }  }); |

### Login

Signal Server에 로그인을 하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| userId | String | 로그인 ID (e-Mail 형식) |
| RoundeeCallback.CallbackResult | callback | 성공/실패 콜백 등록 |

* 예제

|  |
| --- |
| **try** {  Roundee.*getInstance*(**this**).Login(**“Test@gmail.com”**, **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason)  **throws** JSONException {  });  } **catch** (Exception e) {  e.printStackTrace();  } |

### regNotiListener

Conference room에 Noti. 메시지 수신을 위한 listener를 등록하기 위해 사용한다. (Noti Type 에 대응하여 코드 작성필요.)

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| RoundeeCallback.NotiResponse | callback | Noti 콜백 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).regNotiListener(**new** RoundeeCallback.NotiResponse() {  @Override  **public void** getMessage(String roomid, Noti noti, SttRoomInfo sttRoomInfo, SttMember sttMember) **throws** JSONException {  } }); |

### CreateRoom

Conference room을 생성하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roonName | String | Conference room name |
| roomId | String | Conference room id(NULL인경우 자동생성) \*단 유니크한 값 필요. (한번이라도 사용된 ID 사용불가) |
| RoundeeCallback.callbackResult | callback | 성공 /실패 콜백 등록 |

CreateRoom 이후, Signal Server로부터 conference room 정보를 수신한다.

|  |  |
| --- | --- |
| **Return** | **Description** |
| roomID | Signal Server가 생성한 혹은 유저가 설정한 RoomID |

* 예제

|  |
| --- |
| **try** {  String roomId = Roundee.*getInstance*(**this**).CreateRoom(**“TestRoom”**, **“TestID1”**  **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason)  **throws** JSONException {  }  });  } **catch** (JSONException e) {  e.printStackTrace();  } |

### getRoomInfo

Conference room 정보 및 user 정보를 조회하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| onlyMember | Boolean | true : 유저 정보 only  false : room 정보 및 유저 정보 |
| roomId | String | 조회할 Conference room Id |
| RoundeeCallback.RoomCallbackResponse | callback | 방정보를 수신할 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).getRoomInfo(**false** , **“RoomID”**,  **new** RoundeeCallback.RoomCallbackResponse() {  @Override  **public void** onResponse(SttRoomInfo sttRoomInfo, SttMemberList sttMemberList) **throws** JSONException {  } }); |

### StartMedia

Device Camera Open 및 Server 와 Media 연결을 위해 사용한다. Conference Room이 이미 존재하거나 생성한 이후에 동작한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| RelativeLayout | layout | Device 카메라 화면을 뿌려줄 layout |
| roomId | String | 연결할 Conference room ID |
| isFrontCamera | Boolean | Device 의 Camera Front / rear 선택 (true : 전면카메라, false : 후면카메라) |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).StartMedia(**relativelayout**, **“RoomID”**, **true** **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### JoinRoom

생성된 Conference Room에 입장하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomID | String | 입장할 Conference room ID |
| diplayname | String | 상대방 화면에 뿌려줄 이름 |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**). JoinRoom(**“RoomID”**, **“MyName”**,  **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### AddViewer

Conference room에 참여한 User 중 원하는 User 화면을 요청하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| RelativeLayout | layout | 선택한 User 화면을 보여줄 layout |
| roomId | String | Conference room ID |
| userId | String | 대상 userId (자기자신일 경우는 ‘Self’) |
| isShowDisplayName | Boolean | 해당 사용자의 DisplayName On/Off |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).AddViewer(relativeLayout, **“RoomID”**, **“UserID”**,  **true** , **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### changeUserViewer

화면에 뿌려지고 있는 유저 화면을 변경하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomId | String | Conference room ID |
| RelativeLayout | layout | 변경을 하고싶은 layout |
| toId | String | 새로 화면에 뿌리고 싶은 UserID |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| **try** { Roundee.*getInstance*(**this**).changeUserViewer(**“RoomId”**,layout , **“UserId”**  ,**new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  }  }); } **catch** (JSONException e) {  e.printStackTrace(); } |

### updateDisplayName

Conference 참여자에게 보여줄 displayname을 변경하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomID | String | Conference room ID |
| displayname | String | 상대방에게 보여줄 displayname |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).updateDisplayName(**"RoomID"**, **"Judy"**,  **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### switchCamera

디바이스의 전면카메라 / 후면카메라를 선택하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| type | String | “Front” / “Back” (전면 / 후면) |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).switchCamera(**"Front"** ,  **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### getScreenLayout

UserID와 매핑되어있는 layout을 요청하기 위해 사용한다. Noti 메시지로 User Info 가 업데이트 되었을 때 해당 유저의 layout을 찾을 때 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| userId | String | UserID |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).getScreenLayout(**"UserId"**); |

### setMediaMute

Conference중에 미디어 스트림 전송을 on/off 하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomID | String | Conference room ID |
| type | String | audio / video |
| isOn | Boolean | on / off |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).setMediaMute(**“RoomId”**, **"video"**, false, **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {    } }); |

### removeViewer

AddViewer로 연결된 User와 연결을 해제하기 위해 사용한다.

Connection만 종료되며, App에서 추가한 layout 처리는 별도로 필요하다. (remove등)

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomID | String | Conference room ID |
| RelativeLayout | layout | 상대방 화면이 보여지고 있는 layout |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).removeViewer(**“RoomId”**, layout,  **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException { } }); |

### InviteUser

생성된 Conference room에 참여자를 초대하기 위해 사용한다. 참여자는 Email로 초대 하며, 기술된 Email로 Server는 초대 Mail을 발송함.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomID | String | Conference room ID |
| userId | String | 초대할 UserID  (eMail 주소, 여러명일땐 ‘,’ 로 구분) |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).InviteUser(**“RoomId”**, **“T1@gmail.com , T2@gmail.com**,  **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### LeaveRoom

참여중인 Conference room을 나가기 위해 사용한다. 해당 사용자만 Conference room에서 나가게되고 해당 Conference room은 종료되지 않고 남아 있는다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomID | String | Conference room ID |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).LeaveRoom(**“RoomId”**, **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### exitConference

Conference room 개설자가 해당 Conference room을 완전히 종료하기 위해 사용한다. 참여중인 사용자에 의지에 상과없이 해당 Conference room은 종료된다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomID | String | Conference room ID |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).exitConference(**“RoomId”**, **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### logout

Signal Server와 접속을 해제하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).logout(**new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### SendChatMsg

참여중인 Conference room 에서 다른참여자들에서 메시지를 보낼 때 사용한다.

Result 값으로 SttChatMsg 값이 들어오면 SttChatMsgArrayList 에 저장하여 사용 할 수 있다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomID | String | Conference room ID |
| chatMsg | String | 보낼 메시지 |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| SttChatMsgList mSttChatMsgList = new SttChatMsgList();  SttChatMsg sttChatMsg = Roundee.*getInstance*(**this**).SendChatMsg(**“RoomId”**, “안녕하세요” **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } });  mSttChatMsgList.sttChatMsgArrayList.add(sttChatMsg); |

### recordingVideo

참여중인 Conference room 영상을 녹화/정지 할 때 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomID | String | Conference room ID |
| type | String | “stop” / “start” |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).recordingVideo(**“RoomId”**, “stop” **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### addBookmarkMsg

Conference Room 녹화중일때 원하는 지점에 북마크를 추가 할 수 있다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomID | String | Conference room ID |
| bookMarkMsg | String | 북마크 설명(이름) |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).addBookmarkMsg(**“RoomId”**, bookMarkMsg **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### upLoadFile

참여중인 Conference room 에서 파일을 전송 할 때 사용한다..

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| roomID | String | Conference room ID |
| fileName | String | 파일 경로 + 파일이름 |
| RoundeeCallback.FileUploadCallback | callback | 파일 업로드 콜백 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).upLoadFile(**“RoomId”**, filename, **new** RoundeeCallback.FileUploadCallback() {  @Override  **public void** onProgress(**int** progress) {  }   @Override  **public void** onResult(SttUploadResult s) {  } }); |

### downLoadFile

참여중인 Conference room 에서 파일 정보를 수신하여 다운로드 할 때 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| fileName | String | 저장할 파일이름 |
| fileUrl | String | 수신된 SttChatMsg 중 fileUrl |
| RoundeeCallback.CallbackResult() | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).logout(**new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {  } }); |

### getCallLog

마지막으로 통화한 기록을 조회하기 위해 사용한다.

* 예제

|  |
| --- |
| SttCallLog callLog = Roundee.*getInstance*(**this**). getCallLog(); |

### setSpeakerMode

Speakek 모드로 변경할 때 사용한다

* Parameter 값이 true 일경우에는 무조건 스피커폰 으로 출력하며, false 일 경우에는 기본 연결된 출력장치로 출력한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| useSpeak | boolean | 스피커폰 사용여부 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**). setSpeakerMode(**true**); |

### setMuteUser

Conference중에 상대방 Audio Stream을 on/off 하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| targetId | String | Target ID(상대방 ID) |
| roomId | String | Conference room ID |
| isOn | Boolean | on / off |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).setMuteUser([Test@gmail.com](mailto:Test@gmail.com) , **“RoomId”**, false, **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {    } }); |

### setDetailVideoQuality

Conference중에 미디어 스트림 전송을 on/off 하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| MaxBitrate | int | MaxBitrate 값 |
| Bitate | int | Bitrate 값 |
| Frame | int | Frame 값 |
| resolution | Point | Resolution 값 |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).setDetailVideoQuality(**2000000**, 5**00000** , **30** , new Point(1280,720) , **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {    } }); |

### setSimpleVideoQuality

Conference중에 Media quality 를 변경하기 위해 사용한다.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| quality | String | “High” / “Medium” / “Low” |
| RoundeeCallback.CallbackResult | callback | 성공 / 실패 콜백 등록 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).setSimpleVideoQuality(**“High”**, **new** RoundeeCallback.CallbackResult() {  @Override  **public void** onResult(**boolean** success, **int** errorcode, String reason) **throws** JSONException {    } }); |

### setDebuggable

라이브러리 Log 파일 출력 여부를 결정한다..(default 는 false)

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| isDebug | boolean | true : 로그출력 , false : 로그 출력안함 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).setDebuggable(true); |

### setDebugMakeFile

라이브러리 Log파일 생성 여부를 결정한다..(default 는 false)

아래 퍼미션을 추가 해야한다.

<**uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"**/>

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| isMake | boolean | true : 파일생성 , false : 파일 생성안함 |

* 예제

|  |
| --- |
| Roundee.*getInstance*(**this**).setDebugMakeFile (true); |

# **Error Code 명세**

Roundee.io에서 발생되는 Error code 에 대한 설명.

|  |  |
| --- | --- |
| **웹소켓 종료코드** | **원인** |
| 1000 normal closure | 정상 종료 |
| 1003 data cannot accepted | 처리불가 데이터 |
| 1009 message too big | 수신불가 크기 |
| 1011 unexpected condition | 서버 내부 오류 |
| 4000 device removed | 장치삭제, 장치 밀어내기 |
| 4001 duplicated devices | 동일장치로 중복접속 |
| 4002 user locked | 가입자 정지 (KT) |
| 4003 user removed | 가입자 삭제 |
| 4004 password changed | 비밀번호 변경 (SKB) |
| 5001 network disconnected | 네트워크 미연결 |
|  |  |
| **서비스 응답코드** | **원인** |
| 200 OK | 성공 |
| 400 Bad Request | Command 파라미터 오류 |
| 403 Forbidden | response 가 없거나 response 불일치 (인증 실패) |
| 404 Not Found | 존재하지 않는 conference |
| 405 Method Not Allowed | 잘못된 Command |
| 406 Not Acceptable | 잘못된 parameter value |
| 407 Proxy Authentication Required | 다이제스트 인증이 필요함. 단말은 nonce에 대한 reponse 를 새로 생성해야함. |
| 408 Request Timeout | 서버내에서 요청 처리시간이 초과된경우 |
| 409 Conflict | 중복된 conference |
| 410 Gone | key 가 xfrm 으로 삭제된 상태일 경우 |
| 412 Conditional Request Failed | mobile : msp 가 없거나 msp 가 일치하지 않는 경우 |
| 417 Expectation Failed | 요청 처리중 예외 발생 |
| 423 Locked | 가입자/고객사 정지 |
| 500 Internal Server Error | 서버 내부 오류 |
| 501 Not Implemented | 미제공 기능 |
| 503 Service Unavailable | as 서버와 접속이 안되는 경우 |
| 505 Version Not Supported | 현재 앱 버전으로 접속할 수 없음 |
| 507 Insufficient Storage | 가입자가 할당받은 용량을 초과한 경우 |
| 600 Busy Everywhere | Concurrent call의 최대값을 초과하는 경우 |
| 603 Decline | 등록되지 않은 장치 |
| Other | 지정되지 않은 오류 |