

*Zoom Android-RTC-Stack*

**Zoom Android-RTC-Stack**

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Date** | **History** | **Author** |
| **Sept 18, 2016** | First version | Zoom Engineering |
| **Jan 18, 2017** | 1. Support to join Webinar with Panelist member;   2. Add option to show/hide thumbnail video while  viewing/starting share in meeting;  3. Add option to hide “Leave Meeting” item in host  side;   1. Add watermark in MobileRTC | Zoom Engineering |
| **Mar 13, 2017** | 1. Add interfaces to call room device directly; | Zoom Engineering |
| **Jun 19, 2017** | 1. Add interfaces to get a list of participants’ profile and status in meeting. | Zoom Engineering |

1



*Zoom Android-RTC-Stack*

**Configuration**

**Minimum supported Android OS version**

Android 4.0 +

**CPU**

armeabi-v7a and above.

For x86, it can be supported via Intel ARM binary translator (Houdini). But your app should not contain other x86 targeted native binaries.

1. In order for the app to run on x86, all the binary files in your app should target on ARM, binary translator will translate them to support x86.
2. If your app contains other binary files that target on x86, you should delete those files to remain only ARM targets. Because on x86 devices, if an app have binary files target on x86, it also should have the file that target on ARM. Because Zoom Android SDK does not contains any x86 binaries, so other files also can't have binaries that target on x86 in order for the SDK to run.
3. The ARM binary translator works at run time, not at compile time.

**API Details**

You can find the API details in **zoom-mobilertc/doc** in the SDK package.

**Deployment**

**Android Studio Project “gradle project”**

To start using Zoom Android SDK in your project.

2



*Zoom Android-RTC-Stack*

* Create a new project with targeted SDK and above.
* Import Modules **zoommobilertc** and **zoomcommonlib** into your project

**ADT Project**

To start using Zoom Android SDK in your project.

Import **zoom-mobilertc** and **zoom-commonlib** as lib projects into your project

**Build example projects**

* Change constants values defined in src/us/zoom/sdkexample/Constants.java to your values.
* How to build example project in zoom-mobilertc-adt.zip?

1. Use Eclipse (with ADT plug-in installed)
   1. Import 3 projects as Android project
   2. Build example project
2. Use ant
   1. Define ANDROID\_SDK\_HOME to your Android SDK root path as a system environment variable;
   2. Open command line tool
      * cd example
      * ant

* How to build example project in zoom-mobilertc-android-studio.zip
  + cd zoom-mobilertc-android-studio
  + gradlew.bat example:assembleDebug

3



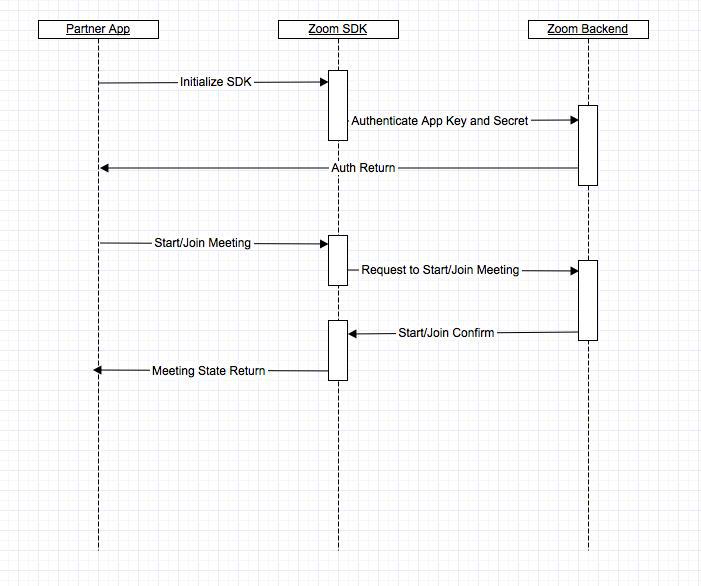
*Zoom Android-RTC-Stack*

**Integration**

We provided a project named Zoom SDK Android Sample. For the detailed usage, please refer to the sample project.

https://github.com/zoomvideo/Android-Sample-Apps

You can find a sample code for each feature in the sdk folder (example, example 2 and loginexample).



4

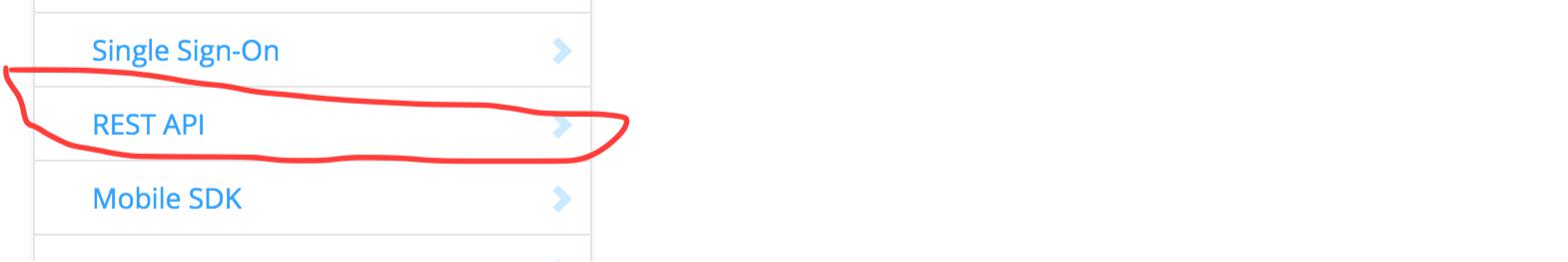


*Zoom Android-RTC-Stack*

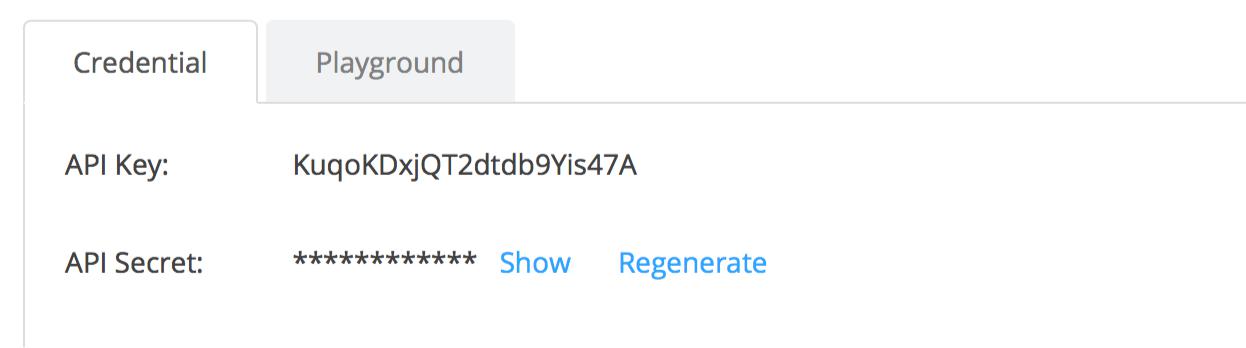
**1. SDK Initialization**

* Get the SDK key and secret from your zoom account. This key/sec is same as the

one used for mobile SDK



* Get the REST API Key/Sec from your zoom account



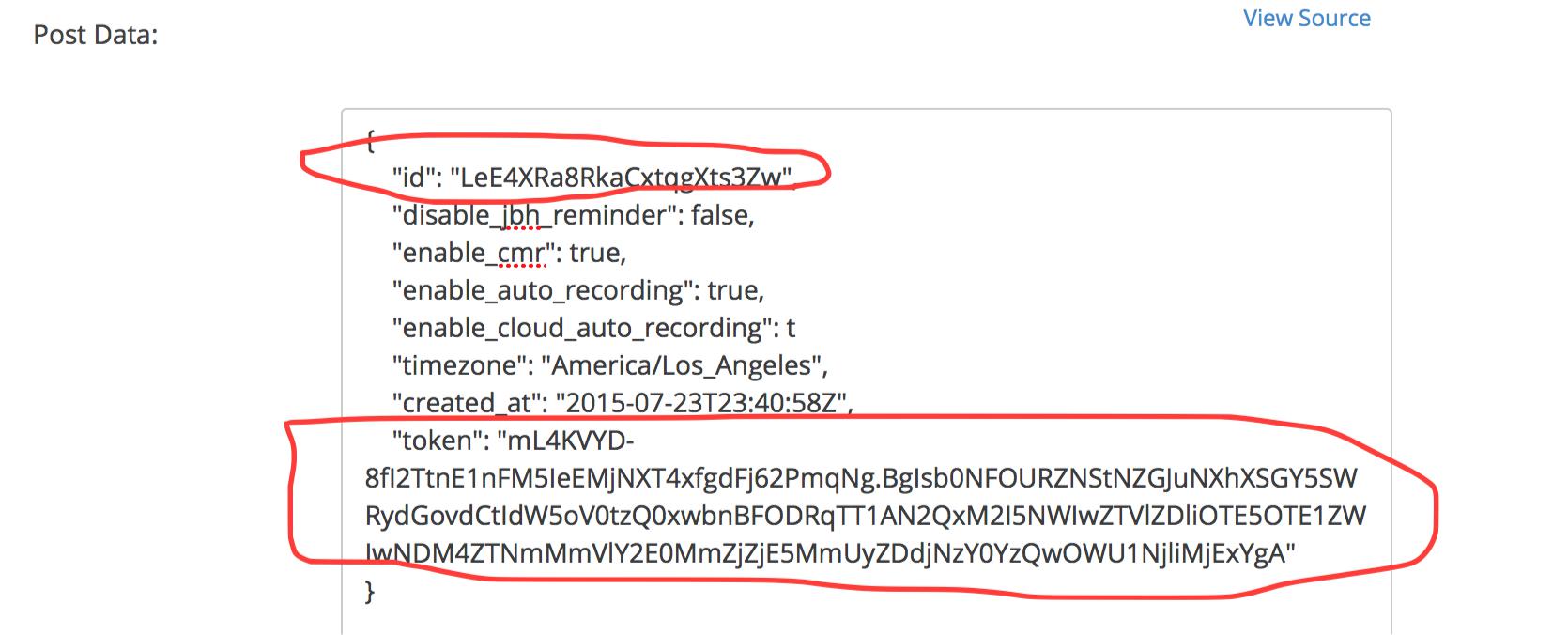
* Call REST API “getbyemail” and this should return the user id and user token

5



*Zoom Android-RTC-Stack*





Here is what SDK initialize code looks like:

ZoomSDK sdk = ZoomSDK.getInstance();

sdk.***initialize***(**this**, *APP\_KEY*, *APP\_SECRET*, *WEB\_DOMAIN*, **this**);

Register ***ZoomSDKInitializeListener*** to get initialize result.

6



*Zoom Android-RTC-Stack*

**2. Login**

There are two options for your app to pass login credentials to the SDK.

1. Use the SDK key/secret from the user account (called SDK user)
2. Pass the Zoom user credentials (login/password) (called login user)

To login using option 2:

ZoomSDK zoomSDK = ZoomSDK.getInstance(); **if**(!(zoomSDK.***loginWithZoom***(userName, password) ==

ZoomApiError.ZOOM\_API\_ERROR \_ SUCCESS))

{ //Error message } else {

//something else

}

For logout:

ZoomSDK zoomSDK = ZoomSDK.*getInstance*(); **if**(!zoomSDK.***logoutZoom***()) {

//Error message } else {

//something else

}

Register ***ZoomSDKAuthenticationListener*** to get login/logout result.

7



*Zoom Android-RTC-Stack*

**3. Start/Join Meeting**

**Create Meeting Service**

ZoomSDK zoomSDK = ZoomSDK.*getInstance*();

**if**(!zoomSDK.isInitialized()) {

//Error message

**return**;

}

MeetingService meetingService = zoomSDK.getMeetingService();

**Start Meeting**

After SDK was initialized, the app can start a zoom meeting.

For API user:

MeetingOptions opts = **new** MeetingOptions();

* *opts.no\_driving\_mode = true;*
* *opts.no\_meeting\_end\_message = true;*
  + *opts.no\_titlebar = true;*
  + *opts.no\_bottom\_toolbar = true;*
* *opts.no\_invite = true;*
* *……………*

**int ret = meetingService.*startMeeting*(this, *USER\_ID*, *ZOOM\_TOKEN*,**

***USER \_TYPE*, meetingNo, *DISPLAY\_NAME*, opts);**

For Login user:

MeetingOptions opts = **new** MeetingOptions();

* *opts.no\_driving\_mode = true;*
* *opts.no\_meeting\_end\_message = true;*
  + *opts.no\_titlebar = true;*
  + *opts.no\_bottom\_toolbar = true;*
* *opts.no\_invite = true;*
* *……………*

**int ret = meetingService.*startMeeting*(this, meetingNo, opts);**

8



*Zoom Android-RTC-Stack*

**Join Meeting**

After SDK was initialized, partner app can join a zoom meeting

ZoomSDK zoomSDK = ZoomSDK.*getInstance*();

**if**(!zoomSDK.isInitialized()) {

//Error message

**return**;

}

MeetingService meetingService = zoomSDK.getMeetingService();

MeetingOptions opts = **new** MeetingOptions();

* *opts.no\_driving\_mode = true;*
* *opts.no\_meeting\_end\_message = true;*
  + *opts.no\_titlebar = true;*
  + *opts.no\_bottom\_toolbar = true;*
* *opts.no\_invite = true;*
* *……………*

**int ret = meetingService.*joinMeeting*(this, meetingNo, *DISPLAY\_NAME*, meetingPassword, opts);**

**Listener for Meeting Service**

Register ***MeetingServiceListener*** to get meeting event.

**4. Pre-Meeting Functions**

After Zoom user is logged in, the app can schedule, edit, delete or get a meeting item

**Create Pre-Meeting Service**

ZoomSDK zoomSDK = ZoomSDK.*getInstance*();

**if**(zoomSDK.isInitialized()) {PreMeetingService preMeetingServic =

zoomSDK.getPreMeetingService();

}

9



*Zoom Android-RTC-Stack*

**Pre-Meeting Service function Api**

publicboolean editMeeting(String meetingId, MeetingItem item);//edit a

meeting

publicboolean scheduleMeeting(MeetingItem item);// schedule a meeting publicboolean deleteMeeting(MeetingItem item); // delete a meeting

publicMeetingItem getMeetingItemByIndex(int index); // get a meeting

item by index

publicMeetingItem getMeetingItemByNumber(long meetingNumber); //get a meeting itemby meeting number

**Listener for Pre-Meeting Service**

Register ***PreMeetingServiceListener*** to get function call result.

1. **Get MeetingSettingsHelper**

**Get Meeting Setting Helper**

ZoomSDK zoomSDK = ZoomSDK.*getInstance*();

**if**(!zoomSDK.isInitialized()) {

//Error message

**return**;

}

MeetingSettingsHelper meetingSettings = zoomSDK.getMeetingSettingsHelper();

10



*Zoom Android-RTC-Stack*

**6. Meeting Options**

You can join a meeting with customized options such as disable driver mode, disable meeting invitation, customize invitation action list , show/hide meeting views etc.

MeetingOptions opts = **new** MeetingOptions();

opts.no\_driving\_mode = true; opts.no\_invite = true; opts.no\_meeting\_end\_message = true; opts.no\_meeting\_error\_message = true opts.no\_titlebar = true; opts.no\_bottom\_toolbar = true; opts.no\_dial\_in\_via\_phone = true; opts.no\_dial\_out\_to\_phone = true; opts.no\_disconnect\_audio = true;

opts.invite\_options = InviteOptions.INVITE\_VIA\_EMAIL +

InviteOptions.INVITE\_VIA\_SMS + InviteOptions.INVITE\_COPY\_URL;

opts.meeting\_views\_options= MeetingViewsOptions.NO\_BUTTON\_SHARE +

MeetingViewsOptions.NO\_BUTTON\_VIDEO;

…

…

**int** ret = meetingService.startMeeting(**this**,*USER\_ID*,*ZOOM\_TOKEN*,

*STYPE*, meetingNo, *DISPLAY\_NAME*, opts);

For details, please see ***MeetingOptions*** and ***MeetingViewsOptions*** class in api docs

11



*Zoom Android-RTC-Stack*

**7. Customize Invitation Method**

Define an activity with intent-filter as below:

<activity android:name=*"us.zoom.sdkexample.MyInviteActivity"*

android:label=*"@string/invite\_acitivity\_name"*

android:icon=*"@drawable/ic\_launcher"* >

<intent-filter>

<action android:name=*"us.zoom.sdkexample.intent.action.MeetingInvite"*

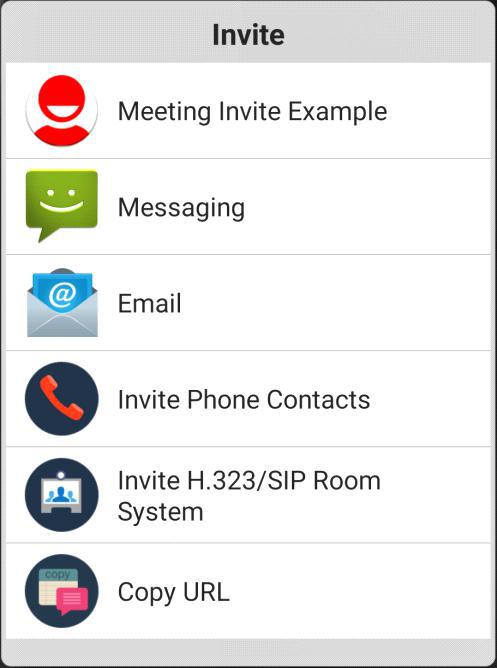
/>

<category android:name=*"android.intent.category.DEFAULT"*

/> </intent-filter> </activity>

The action name should be your application package name plus “*.intent.action.MeetingInvite*”. This activity even can be defined in another application.

Then the activity will be listed at the top of the invite methods list.



Click the item that will open the activity with join meeting URL, invitation topic and invitation content as arguments in the intent. Retrieve them as below:

Intent intent = getIntent(); Uri uri = intent.getData();

String subject = intent.getStringExtra(AndroidAppUtil.*EXTRA\_SUBJECT*); String text = intent.getStringExtra(AndroidAppUtil.*EXTRA\_TEXT*);

12



*Zoom Android-RTC-Stack*

You can also define this bool value in config.xml to remove all default invite options and remain only your totally customized invite activity.

<bool name="zm\_config\_invite\_by\_only\_action\_meeting\_invite">true</bool>

**8. Customize Joining Before Host View**

Define an activity with intent-filter as below:

<activity android:name=*"us.zoom.sdkexample.MyWaitJoinActivity"*

android:icon=*"@drawable/ic\_launcher"* >

<intent-filter>

<action

android:name=*"us.zoom.sdkexample.intent.action.JoinBeforeHost"* />

<category android:name=*"android.intent.category.DEFAULT"*

/> </intent-filter>

</activity>

The action name should be your application package name plus “*.intent.action.JoinBeforeHost*”. Meeting topic, meeting id, time and meeting type ( is repeat or not ) can retrieve from the intent arguments. Retrieve them as below:

Intent intent = getIntent();

Uri uri = intent.getData();

String topic = intent.getStringExtra(AndroidAppUtil.*EXTRA\_TOPIC*); long

meetingId = intent.getLongExtra(AndroidAppUtil*.EXTRA\_MEETING\_ID*, 0);

Finish your custom waiting join activity when the meeting is ready to join.

Register ***MeetingServiceListener*** and implement callback method (***onMeetingEvent***) , handler the event MeetingEvent.MEETING\_READY\_TO\_JOIN to return meeting.

**9. Customize Invitation email subject and content, SMS**

**content, Copy URL text**

Add your invitation content generator class which must implement interface: *com.zipow.videobox.util.InviteContentGenerator*

13



*Zoom Android-RTC-Stack*

Override and implement genEmailTopic and getEmailContent method and set your Email subject and content as the method return value.

Override and implement genSmsContent method and set your Sms content as the method return value.

Override and implement genCopyUrlText method and set your copy url content as the method return value.

If the return value is null or empty, it will use the zoom default invitation template.

Four input parameters have been provided, you can use them to generate your subject and content:

*Application’s context, meeting’s id, meeting url ,user screen name and meeting password*

Add a string resource named “*zm\_config\_invite\_content\_generator*” and set your content generator class full name as the string’s value.

For detail, you can see the “My*InviteContentGenerator*” and config.xml in example

**10. Embed Meeting UI into another Activity**

The example2 in Zoom Android SDK package shows how to embed Zoom meeting UI into another activity.

As you can see in example2, in order for MyMeetingActivity to run, you should put this into a separate process. You can look at example2's AndroidManifest.xml. MyMeetingActivity can not be sub activity of other activities, it needs to bounded at the most outside of other activities. Other activities can be sub activities of MyMeetingActivity. The process where MyMeetingActivity is located will auto-restart for several situations, for example: disconnection of meeting service, crashing of meeting service. If there is no meeting, there is also no meeting process, so MyMeetingActivity also won't start. **you have to follow this rule in your application** **- otherwise Zoom SDK can’t handle it.**

1. **Get in meeting service.**

If code is running in meeting process, InMeetingService object and functions will be reachable. You can get a InMeetingService object like below:

**Get In-Meeting Service**

14





*Zoom Android-RTC-Stack*

ZoomSDK zoomSDK = ZoomSDK.*getInstance*();

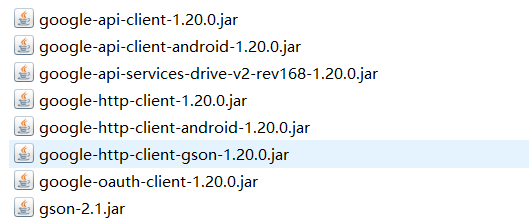
nMeetingService inMeetingService = zoomSDK.getInMeetingService();*//Will return null if the code doesn’t run in meeting process.*

**Listener for In-Meeting Service**

Register ***InMeetingServiceListener*** to get in meeting function call result. Note: *All the callbacks will be reachable in meeting process.*

1. **Enable sharing file from google drive in meeting.**

**Dependency JARS:**



**Goolg Auth Process:**

- **Call ZoomSdk interface to set your google android appliction client id and redirect url**

ZoomSDK sdk = ZoomSDK.*getInstance*();

sdk.~~setGoogleDriveInfo~~(this, CLIENT\_ID

,REDIRECT\_URI);

**CLIENT\_ID, REDIRECT\_URI:**

15



*Zoom Android-RTC-Stack*

Register your application on Google API console and get your android application client ID and redirect uri from: <https://code.google.com/apis/console>

Zoom sdk use the information to get the Google Credential by authorize code.

**Redirect url** : For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorization code for access. From: <https://developers.google.com/identity/protocols/OAuth2InstalledApp>

The Redirect url will be **com.example.app:redirect\_uri\_path**

**com.example.app** is your android application packcage name

**redirect\_uri\_path** just set as /oauth2redirect.

- **Define GoogleAuthactivity in your project manifest to handle the url scheme**

<activity

android:name=*"com.zipow.google\_login.GoogleAuthActivity"*

android:configChanges=*"orientation|screenSize"*

android:launchMode=*"singleTask"* >

<intent-filter>

<!-- Change to your url scheme-->

<data android:scheme=*"***com.example.app***"* />

<action android:name=*"android.intent.action.VIEW"* />

<category android:name=*"android.intent.category.BROWSABLE"* />

<category android:name=*"android.intent.category.DEFAULT"* />

</intent-filter>

</activity>

16