

Android MeetingSDK v0.5.14 Release Notes. (April 9, 2023)

NOTE:

Version 0.5.x of the Visionable Android MeetingSDK is a transitional release that begins to expose functionality available in Visionable's "V3" server architecture. As this architecture is not expected to be in production until 1Q2023, any 0.5.x releases of the MeetingSDK are likely to be unstable and change frequently.

If you are looking to write an application that interfaces with Visionable's V2 architecture, you should remain with v0.4 of the MeetingSDK.

We expect that stability will be established against the V3 architecture with v0.6 of the SDK due when the V3 architecture officially goes into production.

OVERARCHING CHANGES (this text present for ALL v0.5.x releases)

Starting with v0.5, the Visionable MeetingSDK has been re-architected to rely on a layer of cross-platform C++-based code to manage the parsing of XML objects coming from our audio/video engine, for establishing model objects representing Meetings and Participants, and for firing "delegate methods/callbacks" notifying your application of changes in state for the current meeting. Prior to v0.5, these functions were individually implemented per-platform supported in that platform's native language. Moving all of this functionality into a common, C++ codebase should result in consistent behavior when dealing with Visionable back-end servers.

CONNECTING TO V3 SERVERS (this text present for ALL v0.5.x releases)

In Visionable's V3 architecture, a special token (referred to as an MJWT token) is required to join a meeting. There are two types of MJWT tokens: a *guest* MJWT token that doesn't correspond to Visionable user and an *authenticated* MJWT token that is obtained by passing a JWT token obtained from Visionable's authentication system (not covered here). To retrieve an MJWT token, use the new `initializeMeetingWithToken` API call *instead of* the original `initializeMeeting` API call (which is used only with V2 servers).

```
public static void initializeMeetingWithToken(  
    String token,  
    String server,  
    String uuid,  
    IInitializeMeetingCompleteCallback initDoneCallback  
)
```

This function still takes a `uuid` and a `server` name but now also takes a `token` parameter that is either `NULL` if you wish to obtain a guest MJWT or it contains a JWT token if you want to obtain an authenticated MJWT.

The completion routine for `initializeMeetingWithToken` now is called with a second parameter (`String`) that contains the MJWT token (guest or authenticated).

Once you obtain an MJWT, you now join the meeting with a call to `joinMeeting` (same call for either V2 or V3 servers):

```
public static int joinMeeting(  
    String name,  
    String userUUID,  
    IJoinMeetingCompleteCallback joinDoneCallback)
```

The `initializeMeetingWithToken` function caches the `server` name you are connecting to, the `uuid` for the meeting and the MJWT needed to connect to the meeting. When calling `joinMeeting`, you only need to pass the `name` of the user to be shown in the meeting as well as the optional `userUUID` to be associated with this user (pass an empty string to have the SDK generate a UUID for this user). This behavior is slightly different than the APIs in the MeetingSDK for non-Android platforms, and the Android MeetingSDK will likely be refactored in the future to behave in a manner similar to other platforms.

Using these two calls will allow you to connect to a V3 meeting. Once connected, all other SDK functionality is the same as with V2 servers.

CONNECTING TO V2 SERVERS

The APIs for connecting to V2 servers are the same as they were for the v0.4 release of the Android MeetingSDK.

See previous release notes in the v0.5.x series for API changes that were new in previous releases. The rest of these release notes pertain only to the v0.5.14 release.

API CHANGES

No changes.

CHANGES/FIXES

Miscellaneous fixes/improvements in the underlying audio and video engines

Fixed a problem where, sometimes, the `VideoInfo` structure associated with a given stream (and sometimes the corresponding `Participant` object) might not have a name/display name associated with it.

Fixed a problem that was causing delays between the time at which the SDK received events from the underlying audio/video engines and the time at which higher level delegate callbacks were being invoked.

KNOWN ISSUES

While the ability to specify a dedicated `Looper` upon which all delegate methods are invoked, the current Android MeetingSDK does not attempt to create a dedicated `Looper` if one is not specified. This will result in all delegate method calls being made on the same thread being used to parse low-level audio and video events coming from our audio/video engine.

The `VideoView.isScreenShare()` API call will likely return `true` for non screen-share video streams if they are being sent at 4K resolution.

The online documentation for the Apple MeetingSDK does not reflect the API changes made in the v0.5.10.