Apple MeetingSDK v0.5.15 Release Notes. (April 12, 2023)

NOTE:
Version 0.5.x of the Visionable Apple 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 func initializeMeetingWithToken(meetingUUID: String,
server: String, token: String?, completion: @escaping
(Bool,String) -> ())
```

This function still takes a `meetingUUID` and a `server` name but now also takes a `token` parameter that is either `nil` 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 joinMeetingWithToken:

```
public func joinMeetingWithToken(server: String, meetingUUID:
String, token: String, userUUID: String = "", name: String,
completion: @escaping (Bool) -> ())
```

This function takes the `server` name you are connecting to, the `meetingUUID` for the meeting, the MJWT in the `token` parameter, an option `userUUID` to be associated with the user (pass an empty string to have the SDK generate a userUUID), and the `name` of the user to be shown in the meeting.

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 have changed slightly.   The initializeMeeting API call now looks like this:

```
public func initializeMeeting(meetingUUID: String, server:
String, completion: @escaping (Bool,String) -> ())
```

The completion routine now is called with a second argument that is the AES256 encryption key used for the meeting.   Previous SDKs just cached this internally, however now you need to receive it from `initializeMeeting` and pass it to the `joinMeeting` call.

The joinMeeting call now requires you to pass all connection parameters.   If the “userUUID” parameter is an empty string, the SDK will generate a guest-based identifier to associate with this participant:

```
public func joinMeeting(server: String, meetingUUID: String,
key: String, userUUID: String = "", name: String, completion:
@escaping (Bool) -> ())
```

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 v0.5.15 release.

API CHANGES

```
public func setLogFile(_ filename: String)
```

Allows you to provide an absolute path name (or path name relative to the directory the executable is launched from) of a file to be used to log IGAudio, IGVideo, CoreMeeting and MeetingSDK log entries.   In the event of a crash, this file should be updated with the latest log entries before the application terminates.

This API should work for both MacOS and iOS, but with iOS the application would need to determine an appropriate absolute path name and then the application code would need to access the file (there is no way to access this file from outside of the iOS Application's sandbox). You *could* enable your application as one that "shares files" and then find the appropriate Documents directory for your application and provide a full path to a file in that directory (then, in theory, a user could access the log file from the finder when the iOS device is connected to the computer).

For MacOS, you need to find a directory that the application is allowed to write to.   The only such directory guaranteed is the Mac's /tmp directory. You may be able to authorize your application to be able to write to other directories in which case you can then specify a file in such a directory for logging.

IMPORTANT: In order to capture log entries from the underlying audio and video engines, you still need to call `MeetingSDK.shared.enableInlineAudioVideoLogging(true).`

NOTE: This logging *will* function even if the MeetingSDK delegate is not set.

CHANGES/FIXES

Miscellaneous fixes/improvements in the underlying audio and video engines

Fixed a problem in the video engine where a local video stream may be momentarily reported as a remote video stream leading to two video streams for the local user.

Added logging file capabilities as discussed in API changes

KNOWN ISSUES

In support of the new threading model, all delegate methods are executed on a serial OperationQueue that is created by the SDK.   Future versions will allow you to specify an OperationQueue that you create (or use the main queue)

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 release nor the logfile changes made in the v0.5.15 release.