

# Zoom iOS MobileRTC

## Revision History

Date	History	Author
July 30, 2015	First Draft	Robust Hu
Sep 06, 2015	1. Add notification for app become active/inactive; 2. Add ignore signal "SIGPIPE"	Robust Hu
Oct 17, 2015	1. Support to customize the content of "Message Invite" and "Copy URL"; 2. Support to enable/disable "Copy URL"	Robust Hu
Nov 03, 2015	1. Add Feature of Pause/Resume Meeting Audio;	Robust Hu
Jan 15, 2016	1. Support to enable/disable "Invite by Email"; 2. Support to customize the content and subject of "Invite by Email"	Robust Hu
Feb 02, 2016	1. Add Interface to turn on/off cloud record;	Robust Hu
Apr 21, 2016	1. Add Interface to check whether meeting is locked;	Robust Hu
Jul 04, 2016	1. Add Interface "onAppShareSplash" in MobileRTCMeetingServiceDelegate for app share; 2. Add Interface "onJBHWaitingWithCmd:" in MobileRTCMeetingServiceDelegate for customizing to show/hide JBH waiting.	Robust Hu
Aug 11, 2016	1. Add Interface for login with work email; 2. Add Interface for login user to schedule/edit/delete/list meeting	Robust Hu
Aug 26, 2016	1. Support to customize strings in MobileRTC; 2. Support to customize images in MobileRTC	Robust Hu
Oct 08, 2016	1. Support to customize Dial-out	Robust Hu

## Main class description

- **MobileRTC**: MobileRTC base class, take charge of initializing MobileRTC, and return the instance of MobileRTCAuthService, MobileRTCPremeetingService, MobileRTCMeetingService and MobileRTCMeetingSettings.
- **MobileRTCAuthService**: MobileRTC authenticate service class, before using MobileRTC, partner app should request authentication with app key and app secret.
- **MobileRTCAuthDelegate**: MobileRTC authenticate delegate class, partner app can get the authentication result by this class.
- **MobileRTCPremeetingService**: MobileRTC pre-meeting service class, once MobileRTC user login with work email, he/she can use this class to schedule/edit/delete/list meeting.
- **MobileRTCPremeetingDelegate**: MobileRTC pre-meeting delegate class, MobileRTC user can the event result of schedule/edit/delete/list meeting.
- **MobileRTCMeetingService**: MobileRTC meeting service class, take charge of starting/joining meeting.
- **MobileRTCMeetingServiceDelegate**: MobileRTC meeting service delegate class, partner app can listen the meeting result and meeting state by the class
- **MobileRTCMeetingSettings**: MobileRTC meeting settings class, take charge of setting/getting the meeting audio/video state.
- **MobileRTCInviteHelper**: MobileRTC customize invite class, partner app can customize the behavior of inviting participant by the class

## Development Description

### Configuration

- iOS version : iOS 7.0 or later
- Support ARC: Yes

### Frameworks

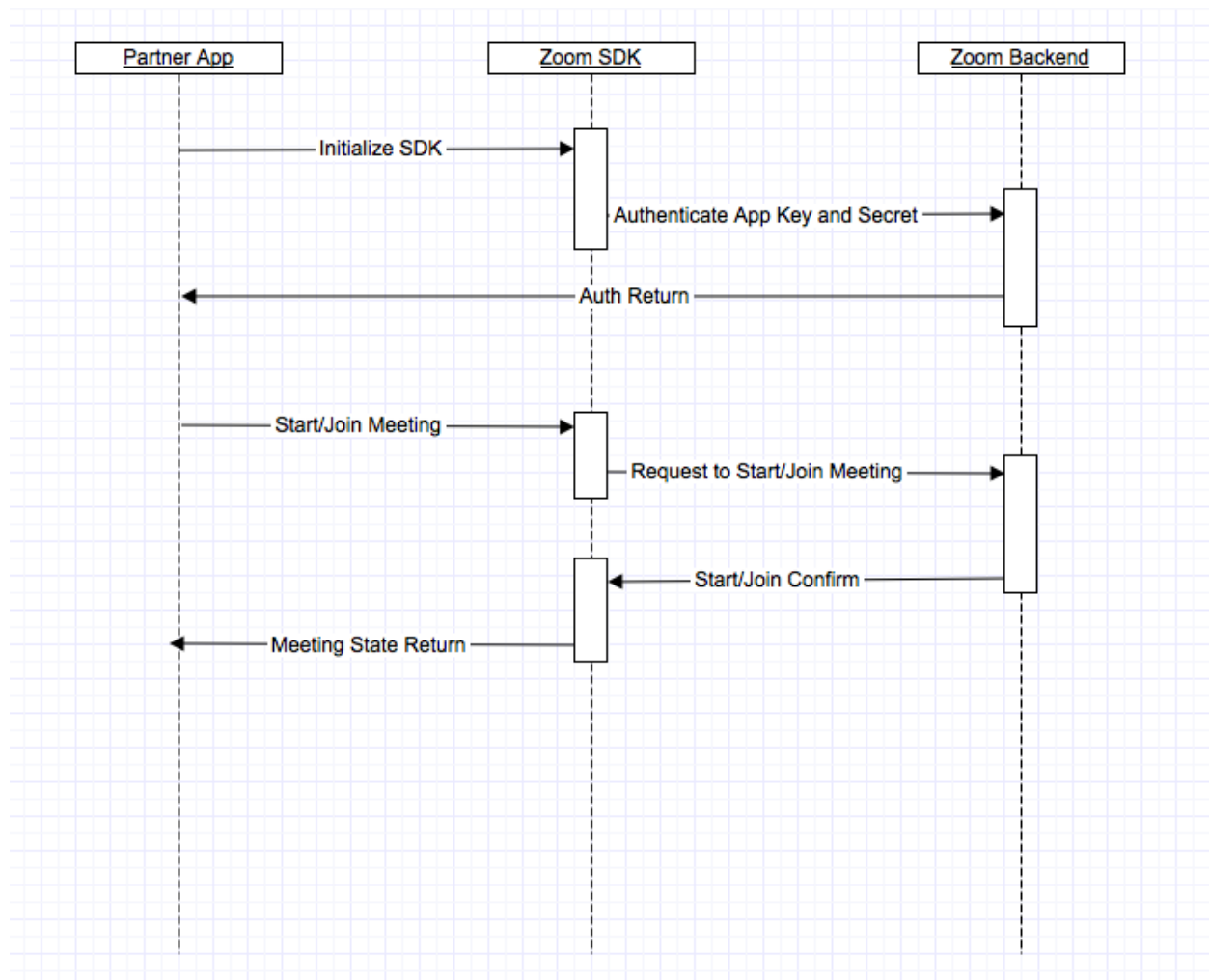
- MobileRTC provides a **MobileRTC.framework** and a **MobileRTCResources.bundle** in zoom-ios-mobilrtc/lib directory;
- MobileRTC.framework should be imported into “Link Binary With Libraries”

- MobileRTC Resources.bundle should be copied into “Copy Bundle Resources”.

## Deployment

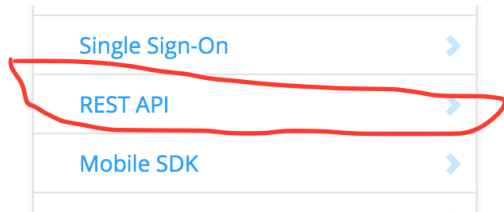
- Set “Targeted Device Family” to “iPhone/iPad”
- Set “iOS Deployment Target” to “iOS 7.0” or later
- Add “-ObjC” in “Other Linker Flags” of Build Settings
- Set “C++ Language Dialect” to “Compile Default”
- Set “C++ Standard Library” to “Compile Default”
- Import “libsqlite3.tbd”, “libstdc++.6.0.9.tbd”, “libz.1.2.5.tbd”, “VideoToolbox.framework” and “CoreBluetooth.framework” into “Link Binary With Libraries”.

## Sequence Diagram



## REST API usages

Get the App key and secret from your zoom account. This key/sec is same as the one used for mobile RTC



Get the REST API Key/Sec from your zoom account

Credential	Playground
API Key:	KuqoKDxjQT2dtdb9Yis47A
API Secret:	***** <a href="#">Show</a> <a href="#">Regenerate</a>

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

## API Playground

API Endpoint:

https://api.zoom.us/v1/user/getbyemail

API Key: \*

KuqoKDxjQT2dtb9Yis47A

API Secret: \*

ylQrZ6uPduFwZ2QpPyhtGawPytNqAEdKYdGd

Data Type:



JSON



XML

[Clear](#)

User Email Address: \*

abc@test.us

Login Type: \*



Work Email



Google



Facebook



SSO



API

[Send API Request](#)

Post Data:

[View Source](#)

```
{
  "id": "LeE4XRa8RkaCxtqgXts3Zw",
  "disable_jbh_reminder": false,
  "enable_cm": true,
  "enable_auto_recording": true,
  "enable_cloud_auto_recording": t
  "timezone": "America/Los_Angeles",
  "created_at": "2015-07-23T23:40:58Z",
  "token": "mL4KVYD-
8fi2TtnE1nFM5leEMjNXT4xfgdFj62PmqNg.Bglsb0NFOURZNSStNZGJuNXhXSGY5SW
RydGovdCtldW5oV0tzQ0xwbnBFODRqTT1AN2QxM2I5NWlwZTVIZDliOTE5OTE1ZW
lwNDM4ZTNmMmVlY2E0MmZjZjE5MmUyZDdjNzY0YzQwOWU1NjliMjExYgA"
}
```

## Code Integration

We provided a project named MobileRTCSample. For the detailed usage, please refer to the sample project.

### 1. MobileRTC Initialize.

To integrate with MobileRTC, MobileRTC should be initialized in [application:didFinishLaunchingWithOptions](#): in AppDelegate class, code snippets as following:

```
//1. Set MobileRTC Domain
[[MobileRTC sharedRTC] setMobileRTCDomain:@"zoom.us"];

//2. Set MobileRTC Root Navigation Controller
//Note: This step is optional, If app's rootViewController is not a UINavigationController, just
ignore this step.
[[MobileRTC sharedRTC] setMobileRTCRootController:navController];

//3. MobileRTC Authenticate
MobileRTCAuthService *authService = [[MobileRTC sharedRTC] getAuthService];
if (authService)
{
    authService.delegate = self;

    authService.clientKey = @"xx-xx-xxx";
    authService.clientSecret = @"yy-yy-yyy";

    [authService sdkAuth];
}
```

**Note: The following is optional**

The following code snippet was used if partner want to login/logout with work email account:

```
//Login with work email account
MobileRTCAuthService *authService = [[MobileRTC sharedRTC] getAuthService];
if (authService)
{
    BOOL ret = [authService loginWithEmail:@"xx@xx.xx" password:@"yyy"];
}

//Logout
MobileRTC AuthService *authService = [[MobileRTC sharedRTC] getAuthService];
if (authService)
{
    [authService logoutRTC];
}
```

## 2. Listener for MobileRTCAuthDelegate.

To check the authentication from MobileRTC, the listener can be added in AppDelegate, code snippets as following:

```
#pragma mark - Auth Delegate

//Sink auth event
- (void)onMobileRTCAuthReturn:( MobileRTCAuthError)returnValue
```

```

{
    NSLog(@"onMobileRTCAuthReturn %d", returnValue);
}

//Sink login event
- (void)onMobileRTCLoginReturn:(NSInteger)returnValue
{
    NSLog(@"onMobileRTCLoginReturn result=%zd", returnValue);

    //Register delegate of pre-meeting service after login with work email account
    MobileRTCPremeetingService *service = [[MobileRTC sharedRTC] getPreMeetingService];
    if (service)
    {
        service.delegate = self;
    }
}

//Sink logout event
- (void)onMobileRTCLogoutReturn:(NSInteger)returnValue
{
    NSLog(@"onMobileRTCLogoutReturn result=%zd", returnValue);
}

```

### 3. Start Meeting

After MobileRTC was initialized, partner app can start a meeting, code snippets as following:

```

MobileRTCMeetingService *ms = [[MobileRTC sharedRTC] getMeetingService];
if (ms)
{
    ms.delegate = self;

    //For API User
    NSDictionary *paramDict = @{@"kMeetingParam_UserID:kSDKUserID,
                                kMeetingParam_UserToken:kSDKUserToken,
                                kMeetingParam_UserType:@( MobileRTCUserType_ZoomUser),
                                kMeetingParam_Username:kSDKUserName,
                                kMeetingParam_MeetingNumber:kSDKMeetNumber,
                                //kMeetingParam_IsAppShare:@(YES)
                                };

    // //For login user start scheduled meeting
    // NSDictionary *paramDict = @{@"kMeetingParam_MeetingNumber:kSDKMeetNumber,
    //                               //kMeetingParam_IsAppShare:@(YES)
    //                               };
    //
    // //For login user start instant meeting

```

```
// NSDictionary *paramDict = @{
//             //kMeetingParam_IsAppShare:@(YES)
//             };

MobileRTCMeetError ret = [ms startMeetingWithDictionary:paramDict];
NSLog(@"onMeetNow ret:%d", ret);
}
```

## 4. Join Meeting

After MobileRTC was initialized, partner app can join a meeting, code snippets as following:

```
MobileRTCMeetingService *ms = [[MobileRTC sharedRTC] getMeetingService];
if (ms)
{
    ms.delegate = self;

    //For Join a meeting with password
    NSDictionary *paramDict = @{
        kMeetingParam_Username:kSDKUserName,
        kMeetingParam_MeetingNumber:meetingNumber,
        kMeetingParam_MeetingPassword:meetingPassword,
        //kMeetingParam_ParticipantID:@"111"
    };

    // //For Join a meeting
    // NSDictionary *paramDict = @{
    //             kMeetingParam_Username:kSDKUserName,
    //             kMeetingParam_MeetingNumber:meetingNumber,
    //             //kMeetingParam_ParticipantID:@"111"
    //             };

    MobileRTCMeetError ret = [ms joinMeetingWithDictionary:paramDict];
    NSLog(@"onJoinaMeeting ret:%d", ret);
}
```

## 5. Listener for Meeting Service

Once Partner App started/joined a meeting, MobileRTC will return the meeting state by Meeting Service Delegate, code snippets as following:

#pragma mark - Meeting Service Delegate

```
- (void)onMeetingReturn:( MobileRTCMeetError)error internalError:(NSInteger)internalError
{
    NSLog(@"onMeetingReturn:%d, internalError:%zd", error, internalError);
}
```



```
- (void)onMeetingStateChange:( MobileRTCMeetingState)state
{
    NSLog(@"onMeetingStateChange:%d", state);
}
```

## 6. Get/Set Meeting Audio/Video State

After MobileRTC was initialized, partner can change the meeting audio/video state, the change will take action in the subsequent meeting, code snippets as following:

```
MobileRTCMeetingSettings *settings = [[MobileRTC sharedRTC] getMeetingSettings];
if (!settings)
    return nil;
```

```
BOOL isAutoConnected = [settings autoConnectInternetAudio];
```

```
BOOL isAudioMuted = [settings muteAudioWhenJoinMeeting];
```

```
BOOL isVideoMuted = [settings muteVideoWhenJoinMeeting];
```

```
BOOL disabled = [settings driveModeDisabled];
```

```
BOOL disabledCallIn = [settings callInDisabled];
```

```
BOOL disabledCallOut = [settings callOutDisabled];
```

```
//Set Meeting Audio/Video State
UISwitch *sv = (UISwitch*)sender;
```

```
[[[MobileRTC sharedRTC] getMeetingSettings] setAutoConnectInternetAudio:sv.on];
```

```
[[[MobileRTC sharedRTC] getMeetingSettings] setMuteAudioWhenJoinMeeting:sv.on];
```

```
[[[MobileRTC sharedRTC] getMeetingSettings] setMuteVideoWhenJoinMeeting:sv.on];
```

```
[[[MobileRTC sharedRTC] getMeetingSettings] disableDriveMode:sv.on];
```

```
[[[MobileRTC sharedRTC] getMeetingSettings] disableCallIn:sv.on];
```

```
[[[MobileRTC sharedRTC] getMeetingSettings] disableCallOut:sv.on];
```

## 7. Pause/Resume Meeting Audio

After started/joined meeting, partner can pause/resume the meeting audio, code snippets as following:

```
BOOL isNoAudio = [[[MobileRTC sharedRTC] getMeetingService] isNoMeetingAudio];  
[[[MobileRTC sharedRTC] getMeetingService] pauseMeetingAudio:!isNoAudio];
```

## 8. Customize to invite participant

Once Partner App started/joined a meeting, Partner app can customize the behavior of inviting participant, code snippets as following:

//To enable to customize Invite Participant, Partner should implement the delegate method “-(void)onClickedInviteButton:(UIViewController\*)parentVC”, Partner can present customized Invite view controller in this method.

```
- (void)onClickedInviteButton:(UIViewController*)parentVC  
{  
    InviteViewController *inviteVC = [[InviteViewController alloc] init];  
    UINavigationController *nav = [[UINavigationController alloc]  
initWithRootViewController:inviteVC];  
    nav.modalPresentationStyle = UIModalPresentationFormSheet;  
  
    [parentVC presentViewController:nav animated:YES completion:NULL];  
}
```

And partner can get meeting ID and join meeting URL by the following properties:

```
NSString *meetingID = [MobileRTCInviteHelper sharedInstance].meetingID;  
NSString *meetingURL = [MobileRTCInviteHelper sharedInstance].joinMeetingURL;
```

## 9. Customize the content of “Invite by Message”, “Copy URL” and “Invite by Email”

Partner App can customize the content of “Invite by Message (SMS)”, “Copy URL” and “Invite by Email” after the meeting is ongoing, and can enable/disable the feature of “Invite by Message”/“Copy URL”/“Invite by Email”, code snippets as following:

```
//For Enable/Disable Copy URL  
[MobileRTCInviteHelper sharedInstance].disableCopyURL = YES;  
  
//For Enable/Disable Invite by Email  
[MobileRTCInviteHelper sharedInstance].disableInviteEmail = YES;  
  
//For Enable/Disable Invite by Message  
[MobileRTCInviteHelper sharedInstance].disableInviteSMS = YES;  
  
if (state == MobileRTCMeetingState_InMeeting)  
{
```

```

        //For customizing the content of Invite by SMS
        NSString *meetingID = [[MobileRTCInviteHelper sharedInstance] meetingID];
        NSString *smsMessage = [NSString stringWithFormat:NSLocalizedString(@"Please join
meeting with ID: %@", @""), meetingID];
        [[MobileRTCInviteHelper sharedInstance] setInviteSMS:smsMessage];

        //For customizing the content of Copy URL
        NSString *joinURL = [[MobileRTCInviteHelper sharedInstance] joinMeetingURL];
        NSString *copyURLMsg = [NSString stringWithFormat:NSLocalizedString(@"Meeting
URL: %@", @""), joinURL];
        [[MobileRTCInviteHelper sharedInstance] setInviteCopyURL:copyURLMsg];

        //For customizing "Invite by Email"
        [MobileRTCInviteHelper sharedInstance].inviteEmailSubject = @"Invite by Email";
        [MobileRTCInviteHelper sharedInstance].inviteEmailContent = [NSString
stringWithFormat:NSLocalizedString(@"Please join meeting with ID: %@", @""), meetingID];
    }

```

## 10. Turn on/off cloud record

```

MobileRTCMeetingService *ms = [[MobileRTC sharedRTC] getMeetingService];
//To check whether support cloud record or not
[ms isCMREnabled];

//To check whether cloud record is in progress
[ms isCMRInProgress];

//Turn on cloud record
[ms turnOnCMR:YES];

//Turn off cloud record
[ms turnOnCMR:NO];

```

## 11. Customize to Dial-Out

Once Partner App started/joined a meeting, Partner app can customize the behavior of Dial-Out if the account supports dial-out, code snippets as following:

```

//To enable to customize Dial-Out, Partner should implement the delegate method “-
(void)onClickedDialOut:(UIViewController*)parentVC isCallMe:(BOOL)me”, Partner can start
dial-out in this method.

- (void)onClickedDialOut:(UIViewController*)parentVC isCallMe:(BOOL)me
{
    MobileRTCMeetingService *ms = [[MobileRTC sharedRTC] getMeetingService];
    if (!ms)

```

```

        return;

    if ([ms isDialOutInProgress])
    {
        NSLog(@"There already exists an ongoing call");
        return;
    }

    NSString *callName = me ? nil : @"Dialer";
    BOOL ret = [ms dialOut:@"+866004" isCallMe:me withName:callName];
    NSLog(@"Dial out result: %zd", ret);
}

```

And partner can get dial-out status by the following code:

```

- (void)onDialOutStatusChanged:(DialOutStatus)status
{
    NSLog(@"onDialOutStatusChanged: %zd", status);
}

```

## 12. Notify app become active/inactive.

MobileRTC should notify the common layer that app will become inactive in [applicationWillResignActive](#): in AppDelegate class, code snippets as following:

```
[[MobileRTC sharedRTC] appWillResignActive];
```

MobileRTC should notify the common layer that app did enter background in [applicationDidEnterBackground](#): in AppDelegate class, code snippets as following:

```
[[MobileRTC sharedRTC] appDidEnterBackground];
```

MobileRTC should notify the common layer that app did become active in [applicationDidBecomeActive](#): in AppDelegate class, code snippets as following:

```
[[MobileRTC sharedRTC] appDidBecomeActive];
```

## 13. Ignore Signal SIGPIPE.

MobileRTC should ignore Signal “SIGPIPE” to avoid crash when app switch from background to foreground, the code can be added in function [main\(int argc, char \\* argv\[\]\)](#) in main.m file, code snippets as following:

```

struct sigaction sa;
sa.sa_handler = SIG_IGN;
sigaction(SIGPIPE, &sa, NULL);

```

## 14. Pre-Meeting Service

This class provides support for schedule/edit/delete meeting once login with work email, code snippets as following:

//For Schedule Meeting

```
MobileRTCPremeetingService *service = [[MobileRTC sharedRTC] getPreMeetingService];
if (service)
{
    id<MobileRTCMeetingItem> item = [service createMeetingItem];
    [item setMeetingTopic:@"xxx"];
    [item setStartTime:[NSDate date]];
    [item setTimeZoneID:[NSTimeZone defaultTimeZone].name];
    [item setDurationInMinutes:60];

    [service scheduleMeeting:item];

    [service destroyMeetingItem:item];
}
```

//For Edit Meeting

```
MobileRTCPremeetingService *service = [[MobileRTC sharedRTC] getPreMeetingService];
if (service)
{
    id<MobileRTCMeetingItem> item = [service getMeetingItemByNumber:123456789];
    if (item)
    {
        [item setMeetingTopic:@"xxx yyy"];
        [item setMeetingNumber:123456789];
        [item setStartTime:[NSDate date]];
        [item setMeetingPassword:@"yyy"];

        [service editMeeting:item];
    }
}
```

//For Delete Meeting

```
MobileRTCPremeetingService *service = [[MobileRTC sharedRTC] getPreMeetingService];
if (service)
{
    id<MobileRTCMeetingItem> item = [service getMeetingItemByNumber:123456789];
    if (item)
    {
        [service deleteMeeting:item];
    }
}
```

//For List Meeting

```
MobileRTCPremeetingService *service = [[MobileRTC sharedRTC] getPreMeetingService];
if (service)
{
    [service listMeeting];
}
```

## 15. Listener for Pre-Meeting Service

Once Schedule/Edit/Delete/List meeting with MobileRTCPremeetingService, MobileRTCMobileRTC will return the event state by Pre-Meeting Service Delegate, code snippets as following:

```
- (void)sinkSchedultMeeting:(NSInteger)result
{
    NSLog(@"sinkSchedultMeeting result: %zd", result);
}

- (void)sinkEditMeeting:(NSInteger)result
{
    NSLog(@"sinkEditMeeting result: %zd", result);
}

- (void)sinkDeleteMeeting:(NSInteger)result
{
    NSLog(@"sinkDeleteMeeting result: %zd", result);
}

- (void)sinkListMeeting:(NSInteger)result withMeetingItems:(NSArray*)array
{
    NSLog(@"sinkSchedultMeeting result: %zd items: %@", result, array);
}
```

## 16. For App Share

iOS MobileRTC supports to start a meeting with sharing content in App(App Share), and at the same time the meeting UI can be hidden in the background while app sharing.

//Once meeting is started, user can sink “onMeetingReady” and start app share in this method

```
- (void)onMeetingReady
{
    MobileRTCMeetingService *ms = [[MobileRTC sharedRTC] getMeetingService];
    if ([ms isDirectAppShareMeeting])
    {
        if ([ms isStartingShare] || [ms isViewingShare])
        {
            NSLog(@"There exist an ongoing share");
            [ms showZoomMeeting:nil];
        }
    }
}
```

```

        return;
    }

    BOOL ret = [ms startAppShare];
    NSLog(@"Start App Share... ret:%zd", ret);
}
}

//Notify that user can share the splash in this method
- (void)onAppShareSplash
{
    MobileRTCMeetingService *ms = [[MobileRTC sharedRTC] getMeetingService];
    if (ms)
    {
        [ms appShareWithView:self.splashVC.view];
    }
}

//Notify that user clicked the share button in meeting UI, user can hide meeting ui and share
his/her content in the App.
- (void)onClickedShareButton
{
    MobileRTCMeetingService *ms = [[MobileRTC sharedRTC] getMeetingService];
    if (ms)
    {
        if ([ms isStartingShare] || [ms isViewingShare])
        {
            NSLog(@"There exist an ongoing share");
            return;
        }

        [ms hideMobileRTCMeeting:^(void){
            [ms startAppShare];
        }];
    }
}

//Notify that there exists no ongoing share, user can determine to share his/her content or not.
- (void)onOngoingShareStopped
{
    NSLog(@"There does not exist ongoing share");
    // MobileRTCMeetingService *ms = [[MobileRTC sharedRTC] getMeetingService];
    // if (ms)
    // {
    //     [ms startAppShare];
    // }
}

```

## 17. Customize waiting UI

When user try to join a meeting which the host has not started, the “Waiting for Host” UI can be shown, and user can customize the waiting UI once received the following callback.

```
//This method is optional
- (void)onJBHWaitingWithCmd:(JBHCmd)cmd
{
    switch (cmd) {
        case JBHCmd_Show:
        {
            UIViewController *vc = [UIViewController new];
            //
            //      NSString *meetingID = [MobileRTCInviteHelper sharedInstance].meetingID;
            //      vc.title = meetingID;
            //
            UIBarButtonItem *leaveItem = [[UIBarButtonItem alloc]
initWithTitle:NSLocalizedString(@"Leave", @"") style:UIBarButtonItemStylePlain target:self
action:@selector(onLeave:)];
            //      [vc.navigationItem setRightBarButtonItem:leaveItem];
            //
            UINavigationController *nav = [[UINavigationController alloc]
initWithRootViewController:vc];
            //      nav.modalPresentationStyle = UIModalPresentationFormSheet;
            //      [self presentViewController:nav animated:YES completion:NULL];
        }
        break;

        case JBHCmd_Hide:
        default:
        {
            [self dismissViewControllerAnimated:YES completion:NULL];
        }
        break;
    }
}
```

### Note:

**Zoom iOS MobileRTC should be integrated in the main thread.**

**For fixing bug that view will move up about 20px after leaving meeting:**

1. Add the following code in method “viewWillAppear” in the view controller which call the interface of starting/joining meeting

```
- (void)viewWillAppear:(BOOL)animated
```



```

{
    [super viewWillAppear:animated];

    //for bug that there exist 20 pixels in the bottom while leaving meeting quickly
    [[UIApplication sharedApplication] setStatusBarHidden:YES
withAnimation:UIStatusBarAnimationNone];
    [[UIApplication sharedApplication] setStatusBarHidden:NO
withAnimation:UIStatusBarAnimationNone];
}

```

2. Keep app continues to run in the background to support audio, add flag in project's info.plist:

```

<key>UIBackgroundModes</key>

<array>

    <string>audio</string>

</array>

```

#### For hiding warning from iOS MobileRTC:

Set "Debug Information Format" = "DWARF" in project's build setting.

#### For customize string resource in iOS MobileRTC:

Partner can define the string resource in Localizable.string file, and the key of string should be matched with the one in MobileRTC. On the other hand, the key and value of defined string should not be the same, or the customized string will not take place the one in MobileRTC.

We customized some strings in MobileRTCSample project, as follows:

```

"Waiting..." = "Sample Waiting...";

"Zoom" = "SDK Sample";

"Leave" = "Exit";
"Leave Meeting" = "Exit Meeting";

```

**For customize image resource in iOS MobileRTC:**

Partner can define the image resource by their own, and the image file name should be equal to the one in MobileRTC, which will take place the one in MobileRTC.

We customized some images in MobileRTCSample project, the name of image files as follows:

icon\_meeting\_share.png

icon\_meeting\_share@2x.png

icon\_meeting\_share@3x.png

icon\_meeting\_stopshare.png

icon\_meeting\_stopshare@2x.png

icon\_meeting\_stopshare@3x.png