**SDK Initialize**

ERTCConfig \*config = [[ERTCConfig alloc] init];

config.nameSpace = **@”YOUR\_NAME\_SPACE”**

config.apiKey = **@”YOUR\_API\_KEY”**;

[[eRTCSDK alloc] initWithConfig:config withSuccess:^(BOOL isValid, NSString \*errMsg) {

//your code

} andFailure:^(NSError \*error) {

//handle error

}];

**User Module:**

* Namespace validation

[eRTCSDK validateNameSpaceWithWorkSpaceName:**@”YOUR\_NAMESPACE”**] withSuccess:^(BOOL isValid, NSString \*errMsg) {

} andFailure:^(NSError \*error) {

}];

* User Login

NSDictionary \*param = @{

@"loginType": **@"email**/**phone"**,

@"appUserId": **@"APP\_USER\_ID"**,

@"password": **@"PASSWORD"**,

@"fcmToken": **@"FCM\_TOKEN"**

};

[[eRTCUserAuthentication sharedInstance] userAuthenticationWithParam:**param** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];

* Change password

NSDictionary \*param = @{

@"loginType": **@"USER\_NAME"**,

@"appUserId": **@"YOUR\_APP\_USER\_ID"**,

@"currentPassword": **@"YOUR\_CURRENT\_PASSWORD"**,

@"newPassword": **@"YOUR\_NEW\_PASSWORD"**

};

[[eRTCUserAuthentication sharedInstance] changePasswordWithParam:**param**

andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];

* Forgot password

NSDictionary \*param = @{@"loginType": **@"YOUR\_USER\_NAME"**,

@"appUserId": **@"YOUR\_APP\_USER\_ID"**};

[[eRTCUserAuthentication sharedInstance] forgotPasswordWithParam:**param** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];

* Get Chat Users

[[eRTCCoreDataManager sharedInstance] fetchChatUserListWithCompletionHandler:^(id ary, NSError \*err) {

}];

* Get Mentioned users

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Get Reaction Users

[[eRTCChatManager sharedChatInstance] getChatReationUserListWithMsgId: **@"MESSAGE\_UNIQUE\_ID"**

andEmojiCode: **@"EMOJI\_CODE"**

andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];

* Logout

[[eRTCAppUsers sharedInstance] logoutUserWithCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];

* User Metadata On – MQTT subscription

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Update Profile

NSDictionary \*param = @{

@"userId" : **@"USER\_ID"**,

@"profileStatus": **@"STATUS\_STRING"**,

@"loginType": **@"email"**

};

[[eRTCAppUsers sharedInstance] updateUserProfileData:**param** andFileData:**PROFILE\_IMAGE\_BASE64\_DATA** andCompletion:^(id json, NSString \* errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];

**Chat Module**

* Get Threads

[[eRTCChatManager sharedChatInstance] getActiveThreads:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];

* Create Thread

NSDictionary \*param = @{

@"recipientAppUserId": **@"RECIPIENT\_APP\_USERID"**,

@"sendereRTCUserId": **@"SENDER\_RTC\_USERID"**,

};

[[eRTCChatManager sharedChatInstance] getChatThreadIDWithParam:**param** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];

* Get Messages

[[eRTCCoreDataManager sharedInstance] getUserChatHistoryWithThreadID:**@”THREAD\_ID”** andCompletionHandler:^(id ary, NSError \*err) {

}];

* Get ChatThread Messages

[[eRTCCoreDataManager sharedInstance] getUserReplyThreadChatHistoryWithThreadID: **@”THREAD\_ID”** withParentID:**@”MsgUniqueId”** andCompletionHandler:^(id ary, NSError \*err) {

}];

* Send Message – It has encryption configured

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Mark as Read

NSDictionary \*param = @{

@"eRTCUserId": **@"ERTC\_USER\_ID"**,

@"msgStatusEvent": **@"MESSAGE\_STATUS"**,

@"sendereRTCUserId": **@"SENDER\_ERTC\_USER\_ID"**

@"msgUniqueId" : **@"MESSAGE\_UNIQUE\_ID"**,

@"tenantId" : **@"TENANT\_ID"**,

@"threadId": **@"THREAD\_ID"**,

@"timeStamp": **TIMESTAMP**

}

[[eRTCChatManager sharedChatInstance] updateMessageWithReadStatus:**param**];

* Message Updates

[[NSNotificationCenter defaultCenter] addObserver:**self**

selector:@selector(**your\_method:**)

name:**DidUpdateChatNotification**

object:nil];

* Message On – MQTT Listener

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Block/unblock user

NSDictionary \*param = @{

@"appUserId": **@"YOUR\_APP\_USERID"**,

@"blockUnblock": **@"block**/**unblock"** // block or unblock

}

[[eRTCAppUsers sharedInstance] ContactblockUnblock:**param** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}] ;

* Mark as favorite

[[eRTCCoreDataManager sharedInstance] isMarkFavouriteWithMessageUniqueId:**@"MESSAGE\_UNIQUEID"** andMarkFavourite:**TRUE**/**FALSE** andCompletionHandler:^(BOOL isMarkFavourite) {

}];

* Get all Favorite Messages

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Get all favorite thread specific messages

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Download Media

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Download Media On – MQTT Listener

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Thread Metadata On

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Send Reaction

[[eRTCChatManager sharedChatInstance] sendTextReactionWithParam**:@"MESSAGE\_UNIQUE\_ID"** andEmojiCode:**@"EMOJI\_CODE"** andEmojiAction:**@"clear"** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

}andFailure:^(NSError \* \_Nonnull error) {

NSLog(@"Error -->%@", error);

}];

* Message metadata On

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Forward chat

**NSArray \*details = @[];**

[[eRTCChatManager sharedChatInstance] ForwardMultiMessageWithParam:**details** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];}];

* Delete Message

NSDictionary \*param = @{

@"msgUniqueId" : **@"MESSAGE\_UNIQUE\_ID"**,

@"chats" : @[ @{@"msgUniqueId" : **@"MESSAGE\_UNIQUE\_ID"**} ],

@"threadId" : **@"THREAD\_ID"**,

@"deleteType": **@"everyone/self"**,

@"msgCorrelationId": **[[NSDate date] timeIntervalSince1970]**

};

[[eRTCChatManager sharedChatInstance] DeleteMessageWithParam:**param** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

[Helper showAlertOnController:@"eRTC" withMessage:error.localizedDescription onController:self];

}];

* Edit Message

NSDictionary \*param = @{

@"message" : **@"UPDATED\_MESSAGE\_STRING"**,

@"threadId" : **@"ThreadID"**,

@"msgUniqueId": **@"MESSAGE\_UNIQUE\_ID"**

};

[[eRTCChatManager sharedChatInstance] editMessageWithParam:**param** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];

* Chat Restore

NSDictionary \*param = @{

@"pageSize": **@10**,

@"currentMsgId": **@"YOUR\_FIRST\_MESSAGE\_ID"** //optional

@"direction": **@"past**/**future"**

};

[[eRTCChatManager sharedChatInstance] loadPreviousChatHistoryWithThreadID**:@"THREAD\_ID"** parameters:param

andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

}andFailure:^(NSError \* \_Nonnull error) {

}];

**Group Module**

* Create Group

NSDictionary \*param = @{

@"groupType" = **@"private"**;

@"name" = **@"YOUR\_GROUP\_NAME"**;

@"participants" = @[

**@"App\_User\_ID"**,

**@"App\_User\_ID"**

];

};

[[eRTCChatManager sharedChatInstance] CreatePrivateGroup:**param** withGroupImage:BASE64\_IMAGE\_DATA andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];

* Get all groups

[[eRTCChatManager sharedChatInstance]getuserGroups:@{} andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

}andFailure:^(NSError \* \_Nonnull error) {

}];

* Get group info by groupId

NSDictionary \*param = @{

@"groupId" : **@"GROUP\_ID"**

}

[[eRTCChatManager sharedChatInstance]getGroupByGroupId:**param** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

}andFailure:^(NSError \* \_Nonnull error) {

}];

* Update Group details

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Add participants

NSDictionary \*param = @{

@"groupId" = **@"GROUP\_ID"**,

@"participants" = **@[**

**@"xxx@yourdomain.com",**

**@"yyy@yourdomain.com",**

**]**

};

[[eRTCChatManager sharedChatInstance]groupAddParticipants:**param** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

} andFailure:^(NSError \* \_Nonnull error) {

}];

* Remove participants

NSDictionary \*param = @{

@"groupId" = **@"GROUP\_ID"**,

@"participants" = **@[**

**@"xxx@yourdomain.com",**

**@"yyy@yourdomain.com",**

**]**

};

[[eRTCChatManager sharedChatInstance]groupRemoveParticipants:param andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

}andFailure:^(NSError \* \_Nonnull error) {

}];

* Add admin

NSDictionary \*param = @{

@"action" = **@"make"**,

@"eRTCUserId" = **@"ERTC\_USER\_ID"**,

@"groupId" = **@"GROUP\_ID"**,

@"targetAppUserId" = **@"APP\_USER\_ID"**

};

[[eRTCChatManager sharedChatInstance]groupmakeDismissAdmin:**param** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

}andFailure:^(NSError \* \_Nonnull error) {

}];

* Remove admin

NSDictionary \*param = @{

@"action" = **@"dismiss"**,

@"eRTCUserId" = **@"ERTC\_USER\_ID"**,

@"groupId" = **@"GROUP\_ID"**,

@"targetAppUserId" = **@"APP\_USER\_ID"**

};

[[eRTCChatManager sharedChatInstance]groupmakeDismissAdmin:**param** andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

}andFailure:^(NSError \* \_Nonnull error) {

}];

)

* Exit group

NSDictionary \*param = @{

@"groupId" = **@"GROUP\_ID"**,

@"participants" = **@[**

**@"CURRENT\_USER\_LOGIN\_ID"**

**]**

};

[[eRTCChatManager sharedChatInstance]groupRemoveParticipants:param andCompletion:^(id \_Nonnull json, NSString \* \_Nonnull errMsg) {

}andFailure:^(NSError \* \_Nonnull error) {

}];

* Remove group pic

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

* Group Metadata On

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

**Typing Indicator Module**

1. Typing indicator subscription

[[NSNotificationCenter defaultCenter] addObserver:**self**

selector:@selector(**your\_controller\_method:**)

name:DidRecievedTypingStatusNotification object:nil];

1. Typing indicator publish

NSDictionary \*param = @{

@"appUserId" = **@"CUREENT\_APP\_USERID"**,

@"eRTCUserId" = **@"CUREENT\_USER\_LOGINID"**,

@"typingStatusEvent" = **@"on**/**off"**

};

[[eRTCChatManager sharedChatInstance] sendTypingStatus:**param**];

**Notification Module**

1. Mute notification

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX

1. Get notification settings

XXXXXXXXXXXXXXXXXX Not available in iOS XXXXXXXXXXXXXXXXXX