TrueConf Mobile SDK

Developer Guide

Table of Contents

1. General Overview

TrueConf Mobile SDK allows you to create custom applications with videoconferencing support on mobile platforms such as Android and iOS/iPadOS. The main advantage of our libraries is the guarantee of high-quality communication in any network and on most mobile devices available on the market. You don't need to worry about adapting video streams to communication channels or the intricacies of hardware encoding to reduce device load and save battery life.

For the convenience of integrating video conferencing into corporate applications, the option to use various integrated development environments (IDEs) is provided:

- Android Studio for Android development
- Xcode for iOS/iPadOS development
- Apache Cordova, React Native, .NET MAUI for cross-platform development

Applications utilizing the SDK require a connection to the corporate TrueConf Server video conferencing server or the TrueConf Online cloud service. To manage users, conferences, and participants, you need to use the TrueConf Server API.

The current version of the SDK is available in the repository `git@git.trueconf.ru:SDK` and needs to be imported as shown below.

1.1. Mobile SDK Features

Currently, the following features are available in TrueConf Mobile SDK:

- Connecting to the selected server
- Authorization with the selected username/password
- Checking another user's status and receiving change notifications
- Call to the specified user
- Receiving an Incoming Call Request
- Join a group conference
- Customization of conference control elements.

At the same time, the capabilities of TrueConf Server for hosting conferences allow you to:

- schedule a conference on the server in advance;
- create conferences using the TrueConf Server API;
- add participants from video conferencing terminals using SIP and H.323 protocols to conferences;
- connect RTSP streams and IP cameras to the conference;
- allow joining the conference via link through a browser (using WebRTC) with a guest account;
- and much more as described in the video conferencing server documentation.

SDK users can also view content streamed from other devices during a conference and freely change the layout of participants on the screen.

The most extensive functionality can be achieved by using the SDK in conjunction with the TrueConf Server API. The SDK allows setting the behavior of a single user (similar to the client application), while the TrueConf Server API enables monitoring the conference status and managing user profiles and permissions overall.

2. Mobile SDK Operation Scheme

TrueConf Server is implemented as a class that provides access to the main functions and a mechanism for subscribing to events. The core functionality for iOS and Android is unified; however, there are some differences. You will learn more about this later.

General operation algorithm:

1. There are a video conferencing server and a client application (terminal).
2. To conduct a video communication session, a user must be assigned to the terminal, performing all operations on their behalf. Guest connections are available.
3. A communication session between two endpoints (terminals) is initiated at the request of one of the participants.
4. For conferences with multiple participants, a separate conference object with its own identifier and lifespan is created on the server (the current version of the SDK does not support this functionality, only connection to an already created group conference is possible). The terminal must either call into such a conference itself or be added to the participant list before it begins.

2.1. Connecting to the server

The terminal, when there is a network connection on the device, attempts to connect to the video conferencing server. Once the connection is established, it remains active for the entire duration of the system's operation. In the event of a disconnection, the system automatically attempts to reconnect. If reconnection is not successful within the set period, the application is notified, and attempts to re-establish the connection continue with reduced intensity.

2.2. Authorization

The terminal must be authorized on the server using a unique user identifier (TrueConf ID) and password. If TrueConf Server has guest licenses available, temporary users with one-time passwords (guests) can be generated.

A single identifier can only be authorized on one terminal, meaning that unlike regular client applications, TrueConf SDK does not support multiple logins. When logging in on another terminal, the previously authorized terminal will be automatically disconnected.

A user is considered authorized on a specific terminal and available for calls as long as the terminal maintains a connection with the server. If the connection is lost, the terminal will attempt to automatically reconnect, and if it fails within a specified time, it is considered offline. Depending on the terminal's settings (auto-login option enabled), upon client restart or connection restoration, it may attempt to authorize using the last used identifier.

In addition to TrueConf ID (login), a display name is used, which appears on interface elements. In some cases, it may be absent, and TrueConf ID will be used instead.

2.3. Conference

The session starts at the initiative of one of the users. For this to happen, the user's application must be connected to the server, and the user must be authorized on it (see general workflow). When a call is made, the terminal enters a waiting mode, which will end either with the start of the conference or with a notification that it cannot be started.

In the first stage, the server verifies that the called party exists and is "online." After this, a request is sent to the called party. It can either be accepted or rejected. If there is no response within a certain period of time, the request is automatically considered rejected. If a positive response is received, the calling party is informed, and the connection between the two participants is established through the server. If both conference participants are in direct view of each other in terms of network topology, a direct connection may be established, bypassing the server.

The session can be terminated either by command from one of the parties or if the broken connection cannot be restored within a specified time.

The conference widget's user interface is embedded in the SDK and cannot be modified except for branding interface elements and manually configuring the video layout. Outside of a conference, the interface and behavior are determined by the application using it.

2.3.1. Video and audio transmission

When connecting to a session, the quality and delay of transmitted video and audio data are determined by the condition of the channel and the performance of the terminals. This is how the SVC technology works, used in the TrueConf video conferencing system. It adapts according to the channel conditions and changes dynamically during the session. There are global maximum stream limitations depending on the type of network connection used (3G, Wi-Fi, wired) and maximum stream limitations set on the server side. When these are present, the option with the minimum bitrate is selected. Bitrate limitation on the terminal side is not supported in the current version of the SDK.

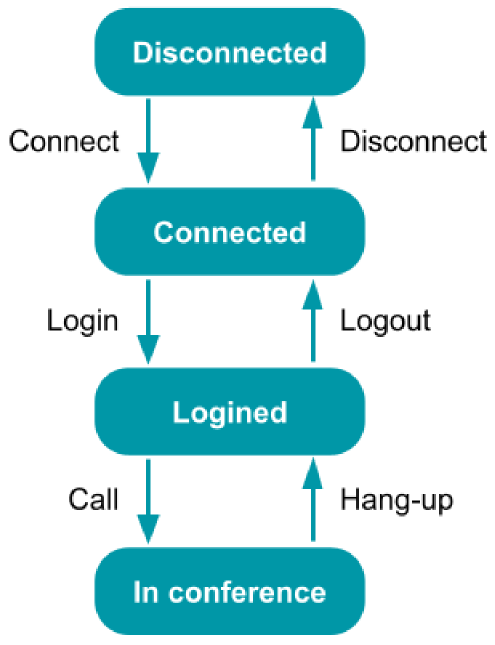
During a session, you can switch between cameras and speakers/headsets. If the channel quality falls below the threshold, it is possible to forcefully disable video sending and receiving, prioritizing the audio stream.

2.4. Additional Service Features

The video conferencing server monitors the status of authorized users and can automatically notify other terminals of any changes (using the onUserStatusUpdate event). Status notifications are based on subscriptions. These can be set up on the server by editing users' address books, or on the terminal by requesting the status of a specific user during a single authorization session.

2.5. Application States with SDK

Operations that lead to changes in terminal mode can take a long time and may be interrupted at any moment (for instance, if the network connection is lost). Therefore, the SDK operates with clearly defined terminal states and transitions between them. The main operations are state change commands and events that report an actual state change or an error that occurred. The state changes are illustrated in the diagram below:



2.6. User statuses

The status of other subscribers, received when subscribing to status changes, differs slightly from the terminal statuses. The correspondence table is provided below.

Terminal Status	User Status	Notes
	Undefined	Technical status, indicates that status information has not yet been received
	Unknown	A subscriber with such TrueConf ID does not exist or their status cannot be retrieved
Disconnected	Offline	

Connected	Offline	
Logged In	Online	The subscriber is registered on the network and can receive incoming calls
In conference	Busy	The subscriber is in a conference and can receive special requests, but a point-to-point call is not possible

2.7. Guest users

You can join public conferences (webinars) as a guest. To do this, generate guest login credentials by following these instructions:

`LOGIN` — `*guest2*EXTID*DISPLAYNAME`

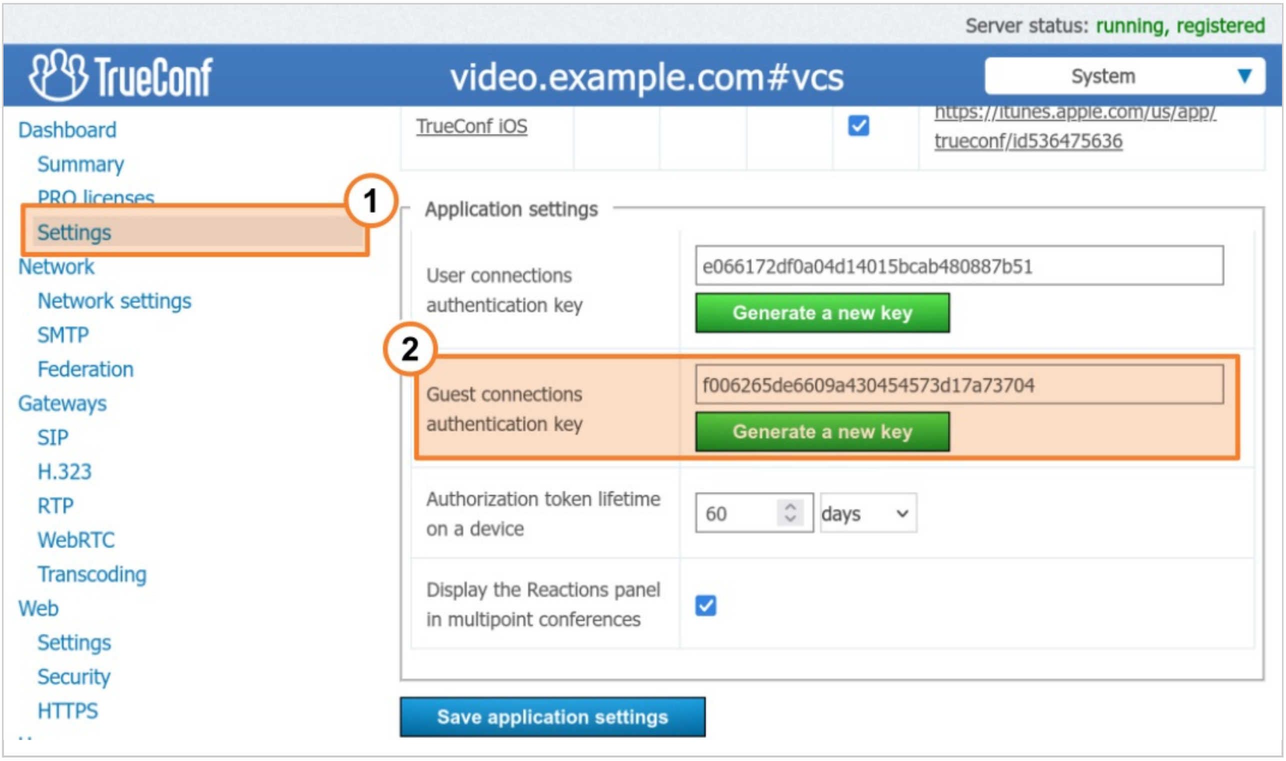
Where:

- `EXTID` — `External ID` (any text)
- `DISPLAYNAME` — the display name for a guest. It can be empty. Must be in UTF8.

`PASSWORD` — `$2RAND*TIMESTAMP*SIGN`

Where:

- `RAND` — any string without the `*` character
- `TIMESTAMP` — the expiration time of the password in UTC (GMT) in seconds since January 1, 1970 (integer)
- `SIGN = md5(RAND + EXTID + TIMESTAMP + SECRET)`, where `SECRET` is the **Guest connections authentication key** parameter of your server from the **Settings** section:



User connections authentication key: e066172df0a04d14015bcab480887b51

Guest connections authentication key: f006265de6609a430454573d17a73704

Authorization token lifetime on a device: 60 days

Display the Reactions panel in multipoint conferences: ✓

If the login begins with `guest2`, the system expects the password in a specific format. If the password is correct and the `TIMESTAMP` is not older than the server time, then upon joining the conference, the guest will have a Call ID of `#guest2:EXTID@server.name`.

> *i* The ability to join public conferences as a guest and the total number of such available connections depend on the license for your TrueConf Server.

3. Accessing and Installing the SDK

The source codes of TrueConf Mobile SDK for mobile applications are located in a remote repository, access to which is restricted and granted by the TrueConf sales department upon your request. In the repository, besides the SDK itself, you will find examples of full-featured applications using the SDK to demonstrate its capabilities.

Below, we will provide detailed instructions on how to access and connect to the repository.

3.1. Step 1. Install Git

We use the Git version control system, so you will need to install software to work with it. Detailed installation instructions for different operating systems can be found on the Git website `git-scm.com`.

3.2. Step 2. Generate an SSH key

As mentioned above, access to the repository is restricted. Git's standard tool, the SSH key, is used for user identification and to grant access.

Such a key is generated by the user and consists of two parts—public and private (stored in two separate files). You provide us with the public part of the key, while you keep the private part on your computer in the special `.ssh` directory. Git will handle the rest. To generate the key, enter the following command in your OS terminal:

```sh
ssh-keygen
```

We recommend using the default folder suggested for installation:

- on Windows OS `C:\Users\MyUser\.ssh`
- on Linux `/home/MyUser/.ssh`
- on macOS `/Users/MyUser/.ssh`

where **MyUser** is your operating system username.

3.3. Step 3. Request a trial period for using the SDK from the sales department

Once the key is generated, contact the sales department to request access to the Git repository. Access to the Android libraries also requires login and password credentials for the Maven repository.

The sales department may inquire about the purposes for which you intend to use the SDK. Please try to describe your project in detail—if we find it interesting, we will provide you with a dedicated technical specialist during the testing period.

Once the sales department decides to grant you access, send them the public part of your key (the file with the `.pub` extension located in the directory where the generated key was stored).

3.4. Step 4. Download the repository

To clone the repository to your computer, execute the following command in your OS terminal:

```sh
git clone git@git.trueconf.ru:SDK TARGET_PATH
```

where `TARGET_PATH` is the path to the directory where you want to copy it.

For example, on Linux, the command might look like this:

```sh
git clone git@git.trueconf.ru:SDK /home/user/mobile-sdk
```

If `TARGET_PATH` is not specified, the repository will be downloaded to the current directory (from which the console is operating).

You can learn about the additional parameters of the `git clone` command on the Git website - https://git-scm.com/docs/git-clone

To avoid working with the console, you can use Git GUI clients with a user-friendly interface, such as:

- GitExtension - https://gitextensions.github.io/
- SourceTree - https://sourcetreeapp.com/

When cloning, use `git@git.trueconf.ru:SDK` as the external repository address.

4. Technical requirements and installation

4.1. xCode – iOS development

You will need Xcode version 12 or later. Supported iOS versions are 13 and later. To continue running the video conference when your app goes into the background, you must enable the appropriate **Background Modes** in your app settings. The `info.plist` file should include the following keys:

```
<key>UIBackgroundModes</key>
<array>
    <string>audio</string>
    <string>voip</string>
</array>
```

For the application to work properly with the camera and microphone, add the following lines to the `info.plist` file:

```
<key>NSCameraUsageDescription</key>
<string>To allow other people to see you</string>
<key>NSMicrophoneUsageDescription</key>
<string>To allow other people to hear you</string>
```

To use the SDK, you need to add the `TrueConfSDK.xcframework` file to your project.

Next, be sure to verify that `TrueConfSDK.xcframework` is registered in your IDE at the following path: **Targets → Project-Name →General →Frameworks, Libraries and Embedded Content**, where **Project-Name** is your project.

Connecting the SDK to the classes of an Objective-C project is done with the following lines:

```
import "TrueConfSDK/TrueConf.h"
import "TrueConfSDK/TrueConfSDK.h"
```

For Swift projects, simply add the line:

```
import TrueConfSDK
```

The framework supports the **arm64** architecture for iOS devices, as well as **x86_64** and **arm64** for simulators.

4.2. Android Studio

The project is designed for use in the Android Studio development environment. The supported Android version is API 24 (Android 7.0) or higher, with targetSdk and compileSdk set to API 34 (Android 14.0). The libraries are compiled using Java 17.

To integrate the SDK, several files in the project need to be modified:

1. In the `.gradle` settings file (learn more about it in the official Gradle documentation):
2. In the `repositories` section, add the Maven repository. The `username` and `password` are provided upon request through your manager:

```
maven {
    credentials {
        username 'username'
        password 'password'
    }
    url 'https://sdk.trueconf.com/maven/repository/maven-public/'
}
```

- In the `dependencies` section, add the SDK libraries of the required version, for example:

```
api 'com.trueconf:trueconfsdk:3.0.0.34@aar'
api 'com.trueconf:media:3.0.0.34@aar'
api 'com.trueconf:jnicore:3.0.0.34@aar'
```

- Additionally, the following dependencies must be specified in the `.gradle` :

```
implementation 'androidx.work:work-runtime:2.9.0'
implementation 'androidx.concurrent:concurrent-futures:1.1.0'
implementation 'androidx.core:core:1.12.0'
implementation 'androidx.appcompat:appcompat:1.6.1'
implementation 'androidx.gridlayout:gridlayout:1.0.0'
implementation 'androidx.mediarouter:mediarouter:1.7.0'
implementation 'androidx.recyclerview:recyclerview:1.3.2'
implementation 'androidx.leanback:leanback:1.0.0'
implementation 'androidx.leanback:leanback-preference:1.0.0'
implementation 'androidx.multidex:multidex:2.0.1'
implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
implementation 'androidx.emoji:emoji-bundled:1.1.0'
implementation 'com.google.android.material:material:1.11.0'
implementation 'org.greenrobot:eventbus:3.3.1'
```

2. Connect the SDK to the project classes in the application code (example for Java):

```java
import com.trueconf.sdk.TrueConfSDK;// to work with SDK methods
import com.trueconf.sdk.TrueConfListener;    // to work with SDK events
```

3. In the `AndroidManifest.xml` manifest file:

- Add the following features to your application:

```
<uses-feature
      android:glEsVersion="0x00020000"
      android:required="true"/>
  <uses-feature
      android:name="android.hardware.audio.low_latency"
      android:required="false"/>
  <uses-feature
      android:name="android.hardware.camera"
      android:required="false"/>
  <uses-feature
      android:name="android.hardware.camera.autofocus"
      android:required="false"/>
  <uses-feature
      android:name="android.hardware.camera.flash"
      android:required="false"/>
  <uses-feature
      android:name="android.hardware.camera.front"
      android:required="false"/>
  <uses-feature
      android:name="android.hardware.microphone"
      android:required="false"/>
  <uses-feature
      android:name="android.hardware.sensor.accelerometer"
      android:required="false"/>
  <uses-feature
      android:name="android.hardware.sensor.light"
      android:required="false"/>
  <uses-feature
      android:name="android.hardware.sensor.proximity"
      android:required="false"/>
  <uses-feature
      android:name="android.hardware.touchscreen"
      android:required="false"/>
  <uses-feature
      android:name="android.hardware.wifi"
      android:required="false"/>
  <uses-feature
      android:name="android.hardware.bluetooth"
      android:required="false"/>
```

- Add the following permissions :

```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
android:maxSdkVersion="32"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

- To enhance the security of network connections, it is recommended to disable the use of the unsecured HTTP protocol. To do this, create a file named `network_security_config.xml` in the `res` directory with the following content:

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
<base-config cleartextTrafficPermitted="false">
    <trust-anchors>
        <certificates src="system" />
    </trust-anchors>
</base-config>
</network-security-config>
```

Then include this XML in the `AndroidManifest.xml` manifest. To do this, add a reference to the created file in the `application` section:

```
<application
        android:networkSecurityConfig="@xml/network_security_config"
        ... >
</application>
```

4.3. .NET

For this framework, TrueConf Mobile SDK is packaged in a NuGet package, which can be integrated using the package manager in Visual Studio. The package includes implementations for both Android and iOS platforms, as well as a common interface for cross-platform development. The libraries are built on **.NET 8** (**net8.0-android** and **net8.0-ios**). Cross-platform MAUI applications are used as examples.

The minimum supported version is Android 7.0 (API 24) and iOS 13.0.

To get started with TrueConf Mobile SDK, you need to install the SDK NuGet package:

1. Right-click on the project name.
2. Select **Add →NuGet packages**.
3. In the window that opens, from the dropdown list of package sources, select **Configure Sources...**.
4. In the packet sources settings, click the **Add** button.
5. In the modal window for adding a source, in the **Location** field, select the directory that contains the `TrueConfSDK.[version].nupkg` file. Enter the name of the source (e.g., **Mobile SDK**), then click **Add Source**.
6. Close the package sources settings to return to the package addition window.
7. In the package source dropdown list in the package addition window, select the added source (for example, **Mobile SDK**). The list will display one package **TrueConfSDK**.
8. Select the **TrueConfSDK** package and click **Add Package**.

Main Objects:

- `ITrueConfSDK` - general interface;
- `TrueConfAndroidSDKImpl` - implementation of the `ITrueConfSDK` interface for Android;
- `TrueConfIOSSDKImpl` - implementation of the `ITrueConfSDK` interface for iOS.

4.3.1. Additional steps for Android

1. Add all permissions and features required for the SDK to the manifest (see item 3 of the **Android** section).
2. Install additional NuGet packages:

```
<PackageReference Include="Xamarin.AndroidX.MediaRouter" Version="1.7.0.4" />
<PackageReference Include="Xamarin.AndroidX.GridLayout" Version="1.0.0.27" />
<PackageReference Include="Xamarin.AndroidX.Leanback.Preference" Version="1.0.0.27" />
<PackageReference Include="Xamarin.AndroidX.Work.Runtime" Version="2.9.0.5" />
<PackageReference Include="Xamarin.AndroidX.Emoji.Bundled" Version="1.1.0.22" />
<PackageReference Include="Xamarin.AndroidX.MultiDex" Version="2.0.1.27" />
```

3. Add the ProGuard obfuscator file with the following content:

```
-dontwarn android.content.**
-dontwarn android.app.**
-dontwarn android.hardware.camera2.**
-dontwarn android.util.**
-dontwarn android.provider.**

-keep class androidx.appcompat.** { *; }
-keep class androidx.startup.InitializationProvider
-keep class com.microsoft.maui.** { *; }
-keep class com.trueconf.sdk.** { *; }
```

4.3.2. Additional steps for iOS

In iOS, you need to add "Privacy - Camera Usage Description" and "Privacy - Microphone Usage Description" to the `Info.plist` file with the text that will be displayed when requesting permission to use the equipment.

4.3.3. Function and Event Mapping for .NET

The description of functions and events can be found in the main SDK documentation; however, their names differ:

Functions		Events	
iOS, Android	.NET	iOS, Android	.NET
stop	Stop	onServerStatus	OnServerStatusEvent
stop	Stop	onServerStatus	OnServerStatusEvent
loginAs	LoginAs	onLogin	OnLoginEvent
logout	Logout	onLogout	OnLogoutEvent
callTo	CallTo	onStateChanged	OnStateChangedEvent
joinConf	JoinConf	onConferenceStart	OnConferenceStartEvent
hangup	Hangup	onConferenceEnd	OnConferenceEndEvent
acceptCall	AcceptCall	onInvite	OnInviteEvent
parseProtocolLink	ParseProtocolLink	onAccept	OnAcceptEvent
scheduleLoginAs	ScheduleLoginAs	onReject	OnRejectEvent
muteMicrophone	MuteMicrophone	microphoneMuted	MicrophoneMuted
muteCamera	MuteCamera	cameraMuted	CameraMuted
getMyId	GetMyId	onRejectTimeOut	OnRejectTimeOutEvent
getMyName	GetMyName	onUserStatusUpdate	OnUserStatusUpdateEvent

isConnectedToServer	IsConnectedToServer	onChatMessageReceived	OnChatMessageReceivedEvent
isLoggedIn	IsLoggedIn	onRecordRequest	OnRecordRequestEvent
isInConference	IsInConference		
getUserStatus	GetUserStatus		
acceptRecord	AcceptRecord		
sendChatMessage	SendChatMessage		

4.4. Cordova

For this framework, TrueConf Mobile SDK is packaged as a Cordova plugin. The minimum supported version for Android is 7.0 (API 24), and for iOS is 13.0.

To get started, create a Cordova project by executing the command:

```sh
cordova create DIRECTORY_NAME PROJECT_PACKAGE DISPLAYED_PROJECT_NAME
```

Then navigate to the created folder and add the TrueConf Mobile SDK plugin using the command:

```sh
cordova plugin add PATH_TO_TRUECONF_PLUGIN
```

No additional actions are necessary to start using TrueConf Mobile SDK. Once the plugin is added to the project, you can obtain an instance of the SDK in js:

```js
let sdkBuilder = cordova.require('trueconf-sdk-plugin.sdk');
  sdkBuilder.getInstance("qa.trueconf.net").then(instance => {
      sdk = instance; // save the SDK instance for further use
  }, error => {
      console.log(error);
  });
```

And then subscribe to and handle the `onStart` event:

```js
sdk.addEventListener(“onStart”, (event) => {
      // now you can use the SDK
  });
```

After that, you can start receiving events and working with the SDK plugin for Cordova.

Static function for obtaining an instance of TrueConf Mobile SDK:

```js
(static) getInstance() → {Promise.<TrueConfSDK>}
```

Returns:

```js
// A promise to be called after starting SDK, it gets the SDK instance.
Type - Promise.<TrueConfSDK>
```

The event signifies the ability to start working with TrueConf Mobile SDK:

```js
onStart
```

Contains parameters:

Name	Type	Description
`connected`	`boolean`	Connection status
`serverName`	`string`	Server name or IP, `undefined` if not connected
`loggedIn`	`boolean`	Authorization status
`userID`	`string`	TrueConf ID, empty if not authorized

4.4.1. Additional steps for Android

In the `build.gradle` file, under the `repositories` section, add the maven repository that contains the Android SDK libraries (as described for Android Studio). The `username` and `password` are provided upon request through your manager.

```
maven {
    credentials {
        username 'username'
        password 'password'
    }
    url 'https://sdk.trueconf.com/maven/repository/maven-public/'
}
```

Before initializing the SDK (prior to calling the `start` method), execute the following methods:

- `registerApp` - you need to pass an `Application` (or its subclass used in the project) to it;
- `setFallbackActivity` – pass the `Activity` class to it, which should be returned to when the call ends.

This can be done, for example, in the `MainActivity` class within `onCreate`:

```js
public class MainActivity extends CordovaActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate();
        TrueConfSDK.getInstance().registerApp(getApplication());
        TrueConfSDK.getInstance().setFallbackActivity(MainActivity.class);
          ...
```

4.5. React Native

For this framework, TrueConf Mobile SDK is packaged in an npm package, which can be added to the project using **npm** or **yarn**.

The minimum supported version is Android 7.0 (API 24) and iOS 13.0.

To create a project, execute the command:

```sh
npx react-native init PROJECT_NAME
```

Then add the SDK module using **npm** or **yarn**:

```sh
npm install PATH_TO_TRUECONF_MODULE --install-links=true
```

or

```sh
yarn add PATH_TO_TRUECONF_MODULE
```

4.5.1. Additional steps for iOS

To connect the SDK module via CocoaPods, run the command from the project directory:

```sh
cd ios && pod install && cd ..
```

Add camera and microphone permissions to `Info.plist` directly in the Xcode project, or by executing the following commands from the project folder:

```sh
cd ios/PROJECT_NAME
plutil -insert NSCameraUsageDescription -string '' Info.plist
plutil -insert NSMicrophoneUsageDescription -string '' Info.plist
```

4.5.2. Additional steps for Android

In `build.gradle` (in `Project:Example`), add the maven repository containing the Android SDK libraries to the `repositories` section for all projects (as for Android Studio). `username` and `password` are provided upon request through your manager.

```
maven {
    credentials {
        username 'username'
        password 'password'
    }
    url 'https://sdk.trueconf.com/maven/repository/maven-public/'
}
```

Before initializing the SDK (before calling the `start` method), it is necessary to call the following methods:

- `registerApp` - you need to pass an `Application` (or its subclass used in the project) to it;
- `setFallbackActivity` - you need to pass the `Activity` class to it, to which the application should return upon the call completion.

This can be done, for example, in the `MainApplication` class in `onCreate` :

```js
public class MainApplication extends Application implements ReactApplication {
    @Override
    public void onCreate() {
        super.onCreate();
        TrueConfSDK.getInstance().registerApp(this);
        TrueConfSDK.getInstance().setFallbackActivity(MainActivity.class);
          ...
```

5. Basic Data Types and Enumerations

5.1. Statuses

The list of user statuses, including the SDK instance itself, is the same across all development platforms, but the names of the constants in the enumeration may differ.

5.1.1. For Android Studio

In Android Studio development, the following statuses are available, provided by the `PresenceStatus` enumeration:

- `UNDEFINED` (-127) — user status information is unavailable
- `INVALID` (-1) — the server does not recognize the user's status
- `LOGOFF` (0) — the user is not connected to the server
- `ONLINE` (1) — the user is authenticated on the server and available for calls
- `BUSY` (2) — the user is in a call or conference
- `MULTIHOST` (5) — the user is in a conference and is the owner. Different types exist to support various kinds of conferences. You can check this with: `status >= BUSY`.

6. Android Functions

Below is a list of features used in development with the Android Studio IDE.

6.1. SDK Instance Functions

They are invoked at the instance level of TrueConf Mobile SDK, obtained via the `TrueConfSDK.getInstance()` function. For example, `TrueConfSDK.getInstance().registerApp(this)`.

6.1.1. registerApp

Registers a subclass of `Application`. This method must be called before `start`.

Parameters:

- `application` – `Application` class (`Class`).

6.1.2. addTrueconfListener

Connecting a participant.

Parameters:

- `listener` – listener class (`TrueconfListener`).

6.1.3. removeTrueconfListener

Muting a participant.

Parameters:

- `listener` – listener class (`TrueconfListener`).

6.1.4. setFallbackActivity

Records the `Activity` class to return to upon call completion. This method must be called without fail.

Parameters:

- `activity` – class `Activity` (`Class`).

6.1.5. start

Launching the SDK.

Parameters:

- `serverList` – list of servers (`String`) (optional parameter);
- `checkPermissions` – whether to check permissions when necessary (`boolean`).

6.1.6. stop

Stopping the SDK instance and releasing resources.

6.1.7. isStarted

Checking the operational status of the SDK instance, result type `boolean`.

6.2. IServerManager

These functions are invoked at the `IServerManager` interface level, which is obtained by calling the `TrueConfSDK.getServerManager()` function.

6.2.1. isConnectedToServer

Checking the connection to the server.

Return value (boolean) - `true` if there is a connection to the server, `false` if there is no connection.

6.2.2. isLoggedIn

Checking authorization status.

Return value (boolean) – `true`, the user is authorized on the server; `false` – the user is not authorized.

6.2.3. loginAs

Authorization as a specific user with the specified parameters on the server.

Parameters:

- `user` - user ID (string);
- `pwd` – password (string);
- `encryptPassword` – `true` if the password is transmitted in plain text (and the SDK needs to encrypt it), `false` if it is already encrypted (boolean);
- `enableAutoLogin` – whether to automatically authenticate the user with this ID upon re-launch (boolean).

Return value (boolean) – `true` if the authorization request is sent to the server, `false` if the authorization cannot be performed.

6.2.4. scheduleLoginAs

Retrieving a list of operations that must be performed sequentially, including the server to connect to, account credentials, and the recipient's call address. This is equivalent to the function `parseProtocolLink`, where parameters are not passed as a protocol string but separately. String parameters may be empty.

Parameters:

- `login` – user ID (string);
- `pwd` – password (string);
- `encryptPassword` – `true` if the password is transmitted in plain text (and the SDK needs to encrypt it), `false` if it is already encrypted (boolean);
- `callToUser` – identifier of the called user (string);
- `autoClose` – indicates whether to close the session with the server after completing a call or interrupting execution (boolean);
- `loginTemp` – indicates a temporary login. It means that the client should be logged out after the call (boolean);
- `loginForce` – forced login. Authentication will occur even if the client is already authenticated on the server (boolean);
- `domain` - the domain in which the automatic server search will be performed (string);
- `serversList` – a list of servers to connect to (string);
- `isPublic` – a parameter indicating that callToUser is a conference name, not a username. If this parameter is incorrect, the call to the user or the connection to the conference will not be executed (boolean).

6.2.5. logout

Deauthorization (log out) of an account without disconnecting the SDK from the video conferencing server.

Return value (boolean) – `true` if the deauthorization request was sent to the server, `false` if deauthorization cannot be performed.

6.3. IConferenceManager

These functions are called at the `IConferenceManager` interface level, which is obtained by calling the `TrueConfSDK.getConferenceManager()` function.

6.3.1. isInConference

Checking conference presence.

Return value (boolean) – `true`, the client is in a conference; `false`, the client is not in a conference.

6.3.2. callTo

Call the specified subscriber.

Parameters:

- `user` – TrueConf ID of the called party (string).

The return value (boolean) is `true` if the call is sent to the server, `false` if the call cannot be made.

6.3.3. joinConf

Joining a group conference.

Parameters:

- `conf_ID` – conference ID (string).

The return value (boolean) is `true` if the call is sent to the server, `false` if the call cannot be made.

6.3.4. hangup

Ending the current call or conference.

Parameters:

- `forAll` (optional) – in the case of a conference, determines whether it should be ended for all participants if authorized (boolean). The default is `true`.

Return value (boolean) — `true` if the call can be ended, `false` if the call cannot be ended (usually due to an incorrect state).

6.3.5. acceptCall

Answering an incoming call.

Parameters:

- `accept` – accept or decline a call (boolean).

Return value (boolean) – `true` if a response to the request can be made, `false` if a response is not possible.

6.3.6. acceptRecord

Response to an incoming video recording request.

Parameters:

- `accept` – accept or decline a video recording request (boolean);
- `userID` – TrueConf ID of the user to whom the response to the request is given (string).

6.3.7. returnToCall

Return to conference activity.

Parameters:

- `currentContext` – current context (`Context`).

6.3.8. sendPincode

Sending a PIN code for access to a PIN-protected conference.

Parameters:

- `confId` – conference ID (`String`);
- `pin` - PIN code (`String`).

6.4. IVideoDeviceController

These functions are called at the `IVideoDeviceController` interface level, which is obtained by calling the `TrueConfSDK.getVideoDeviceController()` function.

6.4.1. isCameraMuted

Camera status check.

Return value (boolean) – `true` when the camera is off, `false` when the camera is on.

6.4.2. muteCamera

Camera status change.

Parameters:

- `mute` – the state to set the camera to: `true` – camera off, `false` – camera on (boolean).

6.4.3. isCameraEnabledByDefault

Returns the default camera status (`boolean`)

6.4.4. setDefaultCameraEnabled

Setting the default camera state.

Parameters:

- `isEnabled` – `true` enables video capture, `false` disables it (`boolean`).

6.5. IAudioDeviceController

These functions are called at the `IAudioDeviceController` interface level, which is obtained by calling the `TrueConfSDK.getAudioDeviceController()` function.

6.5.1. muteMicrophone

Changing the microphone status.

Parameters:

- `mute` – the state to set the microphone: `true` – microphone is off, `false` – microphone is on (boolean).

6.5.2. isMicrophoneMuted

Microphone status check.

Return value (Boolean) – `true` when the microphone is muted, `false` – when the microphone is unmuted.

6.5.3. muteSpeaker

Turns the speaker sound on or off.

Parameters:

- `mute` – `true` mutes the sound, `false` unmutes (`boolean`).

6.5.4. requestAudioState

Requests the current status of audio devices. In turn, calls the onAudioDeviceResponse method from AudioDeviceCallback.

6.5.5. isMicEnabledByDefault

Returns the default microphone status (`boolean`).

6.5.6. setDefaultMicEnabled

Setting the default microphone status.

Parameters:

- `isEnabled` – `true` enables audio capture, `false` disables it (`boolean`).

6.5.7. isSpeakerEnabledByDefault

Returns the default speaker status (`boolean`).

6.5.8. setDefaultSpeakerEnabled

Setting the default speaker status.

Parameters:

- `isEnabled` – `true` enables audio output, `false` disables it (`boolean`).

6.5.9. setDefaultAudioDevice

Setting the default audio output device.

Parameters:

- `audioDeviceInfo` – output device (`AudioDeviceInfo`).

6.5.10. changeAudioDevice

Changes the audio output device during the conference.

Parameters:

- `audioDeviceInfo` – output device (`AudioDeviceInfo`).

6.6. IChatManager

These functions are called at the `IChatManager` interface level, which is obtained by calling the `TrueConfSDK.getChatManager()` function.

6.6.1. sendChatMessage

Sending a text message.

Parameters:

- `toID` – (string) the TrueConf ID of the user to whom the message is being sent (it is recommended to use the full ID in the format `user@server.name`). To send a message to the chat of the current group conference, this parameter should be left empty.
- `text` – (string) message text.

6.7. IContactsManager

These functions are invoked at the `IContactsManager` interface level, which is obtained by calling the `TrueConfSDK.getContactsManager()` function.

6.7.1. getMyId

Getting your own ID.

Return value (string) – the identifier of the current user in the system. If the user is not authorized on the server, Nil is returned.

6.7.2. getMyName

Obtaining a custom name for display in the interface.

Return value (string) – the name of the current user. Contains Nil if the user is not authenticated on the server, or a string equal to the user ID if the user does not have a specific name.

6.7.3. getUserStatus

Retrieving another user's status. If the status is known, it is returned immediately. If not, the status is requested from the server, and the client subscribes to receive notifications of any changes.

Parameters:

- `user` – the TrueConf ID of the user whose status is being requested (string).

Return value (`UserPresStatus`) – the current status of the user.

6.7.4. getUsers

Retrieving the list of contacts from the address book of the user authorized in the SDK. Returns an array of `ContactInfo` objects, each containing the `userId` (String, TrueConf ID of the user) and their current status (`PresenceStatus`).

6.8. IVisicallManager

These functions are invoked at the `IVisicallManager` interface level, which is obtained by calling the `TrueConfSDK.getVisicallManager()` function.

6.8.1. parseProtocolLink

The function receives a command as a string, containing instructions on which account to authenticate and which call to make. It then automatically executes all these operations. If execution of the command is halted at any stage.

Parameters:

- `cmd` - the string being processed (string).

6.9. IExtraButtonController

These functions are called at the `IExtraButtonController` interface level, which is obtained by calling the `TrueConfSDK.getExtraButtonController()` function.

6.9.1. setNewExtraButtons

Add additional buttons to the conference control panel. The buttons will be added in the order they appear in the passed array to the list that opens when tapping the "ellipsis" button (the right button on the panel). See Example 5.

Parameters:

- `btns` – an array of objects of type `TCExtraButton` .

6.10. ICallScreenController

These functions are invoked at the `ICallScreenController` interface level, which is obtained by calling the `TrueConfSDK.getCallScreenController()` function.

6.10.1. setReciveCallFragment

Override the incoming call screen.

Parameters:

- `fragment` – custom fragment of an incoming call (`Fragment`)

6.10.2. setPlaceCallFragment

Override the outgoing call screen.

Parameters:

- `fragment` – custom fragment of an outgoing call (`Fragment`)

6.10.3. setConferenceFragment

Override the conference screen.

Parameters:

- `fragment` – custom conference fragment (`Fragment`)

7. Events for Android

These events are triggered at the `TrueConfListener` interface level (see Example 7). It, in turn, contains the different interfaces listed below, which need to be implemented in your class during development.

7.1. ServerStatusEventsCallback

7.1.1. onServerStatus

Event triggered when a server connects/disconnects or when an error occurs during a connection attempt.

Parameters:

- `connected` – indicates whether there is a connection to the server (boolean);
- `serverName` – the name of the current server (string);
- `serverPort` – the port number used for connecting to the server (numeric).

7.1.2. onStateChanged

Event triggered when the user's own status changes. The current status can be obtained through functions that query the current state.

7.2. LoginEventsCallback

7.2.1. onLogin

Event triggered upon authentication or authentication error on the server.

Parameters:

- `loggedIn` – indicates whether the user is authenticated on the server (boolean);
- `userId` – the user's TrueConf ID (string).

7.2.2. onLogout

Event triggered upon deauthorization on the server.

7.3. ConferenceEventsCallback

7.3.1. onConferenceStart

Event triggered when the conference starts.

7.3.2. onConferenceEnd

Event triggered upon conference completion.

7.3.3. onInvite

Event triggered upon receiving an incoming call.

Parameters:

- `userId` – TrueConf ID of the calling user (string);
- `userName` – the name of the calling user (string).

7.3.4. onAccept

Event triggered when the called party receives the call.

Parameters:

- `userId` – TrueConf ID of the called user (string);
- `userName` – the name of the called user (string).

7.3.5. onReject

Event triggered when the callee rejects the call.

Parameters:

- `userId` – TrueConf ID of the called user (string);
- `userName` – the name of the called user (string).

7.3.6. onRejectTimeout

The event triggered when the called party does not respond within a specified time.

Parameters:

- `userId` – TrueConf ID of the called user (string);
- `userName` – the name of the called user (string).

7.3.7. onRecordRequest

Event triggered upon receiving a request for video recording.

Parameters:

- `userID` – the TrueConf ID of the user requesting the video recording (string);
- `userName` – the name of the user requesting the video recording (string).

7.3.8. onConferencePasswordRequired

Invoked when a PIN code is requested during connection to a conference with a PIN code.

Parameters:

- `confId` – conference ID (`String`)

7.3.9. onConferenceWrongPassword

Triggered when an incorrect PIN is entered while attempting to join a PIN-protected conference.

Parameters:

- `confId` – Conference ID (`String`)

7.4. ChatEventsCallback

7.4.1. onChatMessageReceived

Event triggered upon receiving a text message.

Parameters:

- `fromID` – the user identifier who sent the message (string);
- `fromName` – the name of the user who sent the message (string);
- `text` – message text (string);
- `toID` – the identifier of the user to whom the message was sent (string).

7.5. UserStatusEventsCallback

7.5.1. onUserStatusUpdate

Event triggered when another user's status changes.

Parameters:

- `userID` - the identifier of the user whose status has changed (string);
- `state` — new user status (`UserPresStatus`).

7.5.2. onContactListUpdate

Called when the list of contacts and their statuses is loaded after the user logs in to the server.

7.6. AudioDeviceCallback

7.6.1. onAudioDeviceChanged

Called when the output device is changed during a conference.

Parameters:

- `playerMute` – speaker status (on/off) (boolean)
- `pair` – output device (`AudioDeviceInfo`).

7.6.2. onAudioDeviceUpdate

Called when the state of the microphone or speaker changes.

Parameters:

- `playerMute` – speaker status (on/off) (boolean)
- `recorderMute` – microphone status (on/off) (boolean)
- `pair` – output device (`AudioDeviceInfo`)

7.6.3. onAudioDeviceResponse

Returns the current state of the audio devices in response to the requestAudioState.

Parameters:

- `playerMute` – speaker status (on/off) (boolean)
- `recorderMute` – microphone status (on/off) (boolean)
- `active` – current output device (AudioDeviceInfo)
- `pairs` – list of available output devices (`List<AudioDeviceInfo>`)

7.7. VideoDeviceCallback

7.7.1. onVideoDeviceUpdate

Called when the camera state changes.

Parameters:

- `videoEnabled` – camera status (on/off) (boolean)

7.8. LayoutCallback

7.8.1. onCalculateCustomLayouts

Method for changing the coordinates and sizes of participants' video windows. See Example 7.

7.8.2. onLayoutApplied

Called when the page completes its animation and becomes active.

8. Changes between SDK versions for Android

8.1. What has changed in version 3.0.0 compared to 2.2.0

1. Methods are now grouped by different interfaces:
2. IServerManager
3. IConferenceManager
4. IChatManager
5. IContactsManager
6. IVisicallManager
7. IVideoDeviceController
8. IAudioDeviceController
9. IExtraButtonController
10. ICallScreenController

For example, it was:

```java
TrueConfSDK.getInstance().joinConf(confId);
TrueConfSDK.getInstance().logout();
```

Changed to:

```java
TrueConfSDK.getConferenceManager().joinConf(confId);
TrueConfSDK.getServerManager().logout();
```

2. Significant changes have been made to audio management:

- methods `isSpeakerMuted` , `getAudioDevices` , `onAudioPairChanged` have been removed;
- To obtain the current list of audio output devices, you need to execute the `requestAudioState` request and implement the `onAudioDeviceResponse` method from the `AudioDeviceCallback` interface. This callback also includes the methods `onAudioDeviceChanged` and `onAudioDeviceUpdate` .
- To change the default output device, you must first execute the `requestAudioState` request, and only then set the device using the `setDefaultAudioDevice` method;
- In the `ConferenceFragment` , `IncomingCallFragment` , and `PlaceCallFragment` , the methods `onSwitchMicApplied` and `onMuteSpeakerApplied` have been removed. Instead, you need to implement the interface `AudioDeviceCallback` ;
- The methods `muteMicrophone` , `muteSpeaker` , `changeAudioDevice` should be used only during a conference; otherwise, they will not work due to a check.

3. Changes in video management:

- added the `onVideoDeviceUpdate` method to notify about camera status changes;
- In the `ConferenceFragment` , `IncomingCallFragment` , and `PlaceCallFragment` fragments, the `onSwitchCameraApplied` method has been removed.
- The `muteCamera` method should be used only during a conference; otherwise, it will not work due to verification.

4. Paths to some important classes have been changed, such as the custom button class, as well as `TrueConfListener` and `CallCast` .

Was:

```java
com.trueconf.sdk.data.TCSDKExtraButton
com.trueconf.sdk.interfaces.TrueConfListener
com.trueconf.sdk.gui.activities.CallCast
```

Changed to:

```java
com.trueconf.sdk.presentation.views.TCExtraButton
com.trueconf.sdk.TrueConfListener
com.trueconf.sdk.presentation.activities.CallCast
```

5. The `onStateChanged` method received the `FSM.State newState` parameter, which returns the new user status (`userOffline` , `userOnline` , `userBusy` , etc.). The list of statuses is provided in the table in the "Types. UserPresStatus" section.

6. Some icon names for customization have been changed:

- Instead of `ic_selfie_icon` , `ic_rotate` is now used. Additionally, a resize icon `ic_minimize_fullscreen` has been added to the self-view video.
- The icons `shape_circle_background_red_pressed` and `shape_circle_background_red` are no longer used;
- Added the `conf_button_back` icon for changing the button background.

See the full list in Example 5 "Interface Customization".

7. Libraries are compiled using Java 17 (previously Java 11).

8. The SDK now includes updated libraries OpenSSL 1.1.1w and SQLite 3.47.0.

9. Drag-and-drop support for personal video on the conference screen is enabled.

10. The versions of some dependencies required for the SDK have been updated. The full list is provided in the Integration with Android Studio section.

8.2. What has changed in version 2.2.0 compared to 1.3.3

1. The Android SDK is now distributed through a Maven repository, accessible via a login and password provided upon request through your manager. You need to add the Maven repository to the repositories section in the Gradle file as follows:

```
maven {
    credentials {
        username 'username'
        password 'password'
    }
    url 'https://sdk.trueconf.com/maven/repository/maven-public/'
}
```

2. To integrate the SDK, you need to add three libraries to **dependencies** in the `.gradle` configuration file:

```
api 'com.trueconf:trueconfsdk:2.2.0.101@aar'
api 'com.trueconf:media:2.2.0.101@aar'
api 'com.trueconf:jnicore:2.2.0.101@aar'
```

3. The minimum Android version is now 7.0 (minSdkVersion 24).

4. Connect the SDK to the project classes using the following lines:

```java
import com.trueconf.sdk.TrueConfSDK;// to work with SDK methods
import com.trueconf.sdk.TrueConfListener;    // to work with SDK events
```

5. Before calling the `start` method, you must call the `registerApp` method and pass a subclass of `Application` to it, for example:

```java
public class TestApp extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        TrueConfSDK.getInstance().registerApp(this);
        TrueConfSDK.getInstance().start("server.name", true);
          ...
    }
}
```

6. You need to call the `setFallbackActivity` method, where you should specify the `Activity` class to return to when the call ends, for example:

```java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TrueConfSDK.getInstance().setFallbackActivity(MainActivity.class);
      ...
    }
}
```

7. It is no longer necessary to include the activity `com.vc.gui.activities.Call`, `com.vc.gui.activities.PermissionActivity`, and service `com.vc.service.ExternalSchemeHelperService` in the manifest file.

8. Customization features have been expanded: it is now possible to place your own video and conference participant videos in a separate Fragment, as well as to change the sizes and coordinates of the video layout for participants. Details are described in the documentation for example 7.

9. Some changes in the methods, namely:

- In the `start` method, the `Context` parameter has been removed.
- Added the `registerApp` method, where a subclass of `Application` must be specified;
- Added the `setFallbackActivity` method, where you need to specify the `Activity` class to return to when the call ends;
- Added methods for managing speakers (`muteSpeaker` , `isSpeakerMuted` , `setDefaultSpeakerEnabled` , `getAudioDevices` , `getCurrentAudioDevice` , `setAudioDevice`);
- `onContactListUpdate` - an event in `TrueConfListener.UserStatusEventsCallback` that occurs when the contact list and their statuses are loaded after the user logs in to the server;
- A new interface has been added - `TrueConfListener.AudioDeviceCallback`, which contains 2 methods - `onAudioPairChanged` (invoked when the output device changes) and `onAudioDeviceListChanged` (invoked when the list of devices changes);
- Added methods `setDefaultAudioEnabled` and `setDefaultCameraEnabled` to set the default state of the microphone and camera, respectively;
- The methods `microphoneMuted` and `cameraMuted` have been renamed to `isMicrophoneMuted` and

`isCameraMuted`, respectively;

- The `trueConfSDKLogEnable` flag, which enabled detailed SDK logs, has been removed (they are now always recorded). Additionally, the methods `startSavingLogs` and `stopSavingLogs` have been removed, as logs are automatically collected in the `./files/logs` subdirectory.

9. iOS Functions

Here is a list of functions used when developing in Xcode IDE for iOS.

9.1. initWithViewController

SDK object initialization function.

Parameters:

- `vc` – used for initializing the view controller (`UIViewController`);
- `serverIP` – the server and port to connect to in the format `<server address>:<server port>` (`NSString`);
- `directConnection` (optional) - use of a direct connection (`BOOL`). Default is `YES` .

9.2. start

Launching SDK after initialization.

9.3. startWithServersList

Starting the SDK after initialization with a list of servers. The addresses from the provided list are iterated through, and connection is made to the first operational server.

Parameters:

- `serversList` – a list of server addresses separated by commas (`NSString`).

9.4. isStarted

Checking the operational status of the SDK instance, result type `boolean` .

9.5. stop

Stopping the SDK instance and releasing resources.

9.6. loginAs

Authorization as a specific user with the specified parameters on the server.

Parameters:

- `user` - user ID (string);
- `pwd` – password (string);
- `encryptPassword` – `true` if the password is transmitted in plain text (and the SDK needs to encrypt it), `false` if it is already encrypted (boolean);
- `enableAutoLogin` – whether to automatically authenticate the user with this ID upon re-launch (boolean).

Return value (boolean) – `true` if the authorization request is sent to the server, `false` if the authorization cannot be performed.

9.7. logout

Deauthorization (log out) of an account without disconnecting the SDK from the video conferencing server.

Return value (boolean) – `true` if the deauthorization request was sent to the server, `false` if deauthorization cannot be performed.

9.8. callTo

Call the specified subscriber.

Parameters:

- `user` – TrueConf ID of the called party (string).

The return value (boolean) is `true` if the call is sent to the server, `false` if the call cannot be made.

9.9. joinConf

Joining a group conference.

Parameters:

- `conf_ID` – conference ID (string).

The return value (boolean) is `true` if the call is sent to the server, `false` if the call cannot be made.

9.10. hangup

Ending the current call or conference.

Parameters:

- `forAll` (optional) – in the case of a conference, determines whether it should be ended for all participants if authorized (boolean). The default is `true`.

Return value (boolean) — `true` if the call can be ended, `false` if the call cannot be ended (usually due to an incorrect state).

9.11. acceptCall

Answering an incoming call.

Parameters:

- `accept` – accept or decline a call (boolean).

Return value (boolean) – `true` if a response to the request can be made, `false` if a response is not possible.

9.12. parseProtocolLink

The function receives a command as a string, containing instructions on which account to authenticate and which call to make. It then automatically executes all these operations. If execution of the command is halted at any stage.

Parameters:

- `cmd` - the string being processed (string).

9.13. scheduleLoginAs

Obtaining a list of operations that need to be executed sequentially, including the server to connect to, account credentials, and the recipient's call details. This is equivalent to the function `parseProtocolLink`, where parameters are not passed as a protocol string but individually. String parameters may be empty.

Parameters:

- `login` – user ID (string);
- `pwd` – password (string);
- `encryptPassword` – `true` if the password is transmitted in plain text (and the SDK needs to encrypt it), `false` if it is already encrypted (boolean);
- `callToUser` – identifier of the called user (string);
- `autoClose` – indicates whether to close the session with the server after completing a call or interrupting execution (boolean);
- `loginTemp` – indicates a temporary login. It means that the client should be logged out after the call (boolean);
- `loginForce` – forced login. Authentication will occur even if the client is already authenticated on the server (boolean);
- `domain` - the domain in which the automatic server search will be performed (string);
- `serversList` – a list of servers to connect to (string);
- `isPublic` - a parameter indicating that `callToUser` is the name of a conference, not a user name. If this parameter is incorrect, the call to the user or connection to the conference will not be successful (boolean).

9.14. muteMicrophone

Changing the microphone status.

Parameters:

- `mute` – the state to set the microphone: `true` – microphone is off, `false` – microphone is on (boolean).

9.15. muteCamera

Camera status change.

Parameters:

- `mute` – the state to set the camera to: `true` – camera off, `false` – camera on (boolean).

9.16. getMyId

Getting your own ID.

Return value (string) – the identifier of the current user in the system. If the user is not authorized on the server, Nil is returned.

9.17. getMyName

Obtaining a custom name for display in the interface.

Return value (string) – the name of the current user. Contains Nil if the user is not authenticated on the server.

9.18. isConnectedToServer

Checking the connection to the server.

Return value (boolean) - `true` if there is a connection to the server, `false` if there is no connection.

9.19. isLoggedIn

Checking authorization status.

Return value (boolean) – `true`, the user is authorized on the server; `false` – the user is not authorized.

9.20. isInConference

Checking conference presence.

Return value (boolean) – `true`, the client is in a conference; `false`, the client is not in a conference.

9.21. getUserStatus

Retrieving another user's status. If the status is known, it is returned immediately. If not, the status is requested from the server, and the client subscribes to receive notifications of any changes.

Parameters:

- `user` – the TrueConf ID of the user whose status is being requested (string).

Return value (`UserPresStatus`) – the current status of the user.

9.22. microphoneMuted

Microphone status check.

Return value (Boolean) – `true` when the microphone is muted, `false` – when the microphone is unmuted.

9.23. cameraMuted

Camera status check.

Return value (boolean) – `true` when the camera is off, `false` when the camera is on.

9.24. acceptRecord

Response to an incoming video recording request.

Parameters:

- `accept` – accept or decline a video recording request (boolean);
- `userID` – TrueConf ID of the user to whom the response to the request is given (string).

9.25. sendChatMessage

Sending a text message.

Parameters:

- `toID` – (string) the TrueConf ID of the user to whom the message is being sent (it is recommended to use the full ID in the format `user@server.name`). To send a message to the chat of the current group conference, this parameter should be left empty.
- `text` – (string) message text.

Return value (boolean) – `true` if the message is sent to the server, `false` if the message fails to send due to lack of server connection.

9.26. setInitViewController

Called before the call interface is launched. Allows you to override the default view controller in the SDK. It takes precedence over the view controller specified during SDK initialization.

Parameters:

- `aInitViewController` – the view controller to be used (`UIViewController`).

9.27. setNewExtraButtons

Add additional buttons to the conference control panel. The buttons will be added to the list that opens when tapping the ellipsis button (the right button on the panel) in the order they appear in the provided array. See Example 5.

Parameters:

- `btns` – an array of objects of type `UIAlertAction` (`NSArray`).

9.28. orientationWillChangeTo

It should be called at the start of the interface orientation change. Event relay for the main application controller.

Parameters:

- `toOrientation` – new interface orientation (`UIInterfaceOrientation`).

9.29. orientationDidChangeTo

Should be called upon completion of the interface orientation change. Event relay of the main application controller.

Parameters:

- `toOrientation` – new interface orientation (`UIInterfaceOrientation`)

9.30. trueConfSDKLogEnable

A `BOOL` type property that enables or disables extended logging. By default, it is set to `NO` .

9.31. getUserName

The function returns the display name of another user.

Parameters:

- `user` – TrueConf ID of the user whose name is being requested (string).

Return value (string) – user name.

10. Changes Between SDK Versions for iOS/iPadOS

10.1. Version 3.4.3

What has changed in version 3.4.3 compared to 3.2.6:

- The properties `hideSelfViewButtonsInSmallSize` and `muteButtonVisible` have been removed due to obsolescence.
- The `setNewExtraButtons` function now takes an array of `UIAlertAction` as a parameter instead of `TCSDKExtraButton`. Custom buttons are now added to the menu that opens by clicking the "ellipsis" button (see Example 5);
- The parameter `confCustomControlsImages` has been removed from the initializer `initWithViewController` due to obsolescence. To replace the images of the standard conference control buttons, simply add images with the corresponding names to `Assets` (see Example 5).

11. Examples for Android

Here are several examples for Android Studio IDE when developing for Android. You can download them from our GitHub.

11.1. Example 1. Demonstration of the main capabilities of the SDK

A single-page application where all the main features of TrueConf Mobile SDK are implemented:

1. Initialization and server connection (`start`).
2. Manual login (by pressing a button) on the specified server with the user's account and manual logout from the account.
3. Call a user by their TrueConf ID (`callTo`).
4. Ability to receive incoming video calls.

In the `TestApp` class within `onCreate()` , we first call the method `registerApp()` , passing a subclass of `Application` . We then launch the SDK using the method `start()` .

In `MainActivity` , within `onCreate()` , we call the method `setFallbackActivity()` specifying the `Activity` class to return to in case the call ends.

In `PlaceholderFragment` , we implement the `TrueconfListener` interface and override its functions to receive `callback` .

11.2. Example 2. Demonstration of working with TrueConf links

Ability to connect to the server, make a call using the `trueconf:` protocol with automatic login, and call a specific user via their TrueConf ID, or join a specific group conference using its ID.

The call is executed using the method `parseProtocolLink` with a call string in the `String` format as its parameter.

You can read more about the `trueconf:` control protocol in this article.

11.3. Example 3. Working with group conferences

One of the key features of TrueConf Mobile SDK is the ability to participate in video conferences with multiple users simultaneously. Currently, it only allows joining existing conferences.

The example is the same as the first one, except that instead of using the `callTo` method to connect to a conference, we use the method `joinConf(conferenceId)` .

11.4. Example 4. Working with User Statuses

The example shows monitoring the statuses of other users on the server.

Example for 2 screens: the first screen is used for connecting to the server and authorization; afterwards, the second screen opens with a list of all users from our address book.

The example also demonstrates how to obtain the current user status using the method `getUserStatus(user)` , as well as how to handle `callback` in `onUserStatusUpdate` and `onContactListUpdate` .

11.5. Example 5. Interface Customization

The example demonstrates how to add buttons to the in-call screen. This is done using the method `setNewExtraButtons` , which requires an array of `TCExtraButton` objects as an argument. The list of buttons will visually appear when the "ellipsis" button is pressed during a conference. As an example, 2 buttons are added, which, when pressed, open a new window over the conference window. The first button opens a `Fragment` , and the second opens an `Activity` .

To replace the icons within a call, upload the resources to the `drawables` folder with the following names:

```
ic_bluetooth_audio_arrow_off
ic_bluetooth_audio_arrow
ic_bluetooth_audio
ic_call_end
ic_camera_off
ic_camera_on
ic_dyn_arrow_off
ic_dyn_arrow
ic_empty_invoice
ic_headphones_arrow_off
ic_headphones_arrow
ic_headset_mic_arrow_off
ic_headset_mic_arrow
ic_headset_mic
ic_headset
ic_mic_off
ic_mic_on
ic_more_horiz
ic_phone_in_talk_arrow_off
ic_phone_in_talk_arrow
ic_phone_in_talk_off
ic_phone_in_talk
ic_sound_off
ic_sound_on
ic_rotate
ic_minimize_fullscreen
conf_button_back
```

11.6. Example 6. Chat

The example demonstrates the capability to use the send text messages function `sendChatMessage` and handle the event for incoming messages `onChatMessageReceived`.

If a message is sent to a user who is offline, it will be delivered as soon as they come online. If a message is sent when there is no connection to the server, it is queued and will be sent once the server connection is restored.

11.7. Example 7. Customizing the video layout in a conference

The example demonstrates the ability to place your own image and the images of conference participants in a separate Fragment. It shows the independent implementation of conference control buttons, equipment setup before the conference begins, as well as replacing incoming/outgoing call windows with custom Fragments.

The features demonstrated in the example and the code snippets used for this purpose:

1. Creating a custom incoming call window based on a subclass of `IncomingCallFragment`. To modify the behavior when accepting or declining a call, override the `onAcceptClick` and `onDeclineClick` methods accordingly.
2. Customization of the outgoing call window based on a subclass of `PlaceCallFragment`. If you need to change the behavior when a call is canceled, override the `onHangupClick` method.
3. Modifying the conference window based on a subclass of `ConferenceFragment`.
4. Resizing the call window using the method `setCallLayoutParams`.
5. Control the microphone using the method `setDefaultMicEnabled()`, etc.
6. Camera management using the method `setDefaultCameraEnabled()`, etc.
7. Speaker configuration using the method `setDefaultSpeakerEnabled()`, among others.
8. Changing the size and coordinates of participants' video windows in a conference.

i To see how the functionality for changing window sizes and coordinates works, uncomment the contents of the `onCalculateCustomLayouts` method in `MainActivity` in the example.