Exotel Voice Client Android SDK Integration Guide

Version	Date	Change Log
0.1	04 Nov 2019	Initial Draft
0.2	20 Dec 2019	Final Draft
0.3	30 Dec 2019	Added Troubleshooting Guide. Misc. Updates.
0.4	11 Feb 2020	Changed voip identifier
0.5	22 Apr 2020	Updated Troubleshooting Guide
0.6	02 May 2022	Updated the latest changes from exotel voice sdk.
1.0	21st June 2022	Review comments
1.1	7th july 2022	Integration document test feedback updated.

1. Introduction

ExotelVoice library enables you to add the voip calling feature into your app. This document outlines the integration steps. The library supports only peer to peer 2-way calls. Multi-party conferencing use cases are not supported.

Section 4 describes integration steps for ExotelVoice Android SDK. Section 5.1 describes the webhooks that should be supported in your application backend for number-masking workflow. Section 5.2.1 describes subscribers provisioning in the Exotel platform.

2. Licensing

Create a trial account in Exotel. To enable voip calling in the trial account, contact *exotel support*. Once Exotel support enables the voip capability in the account , a VOIP Exophone will be created as shown below and available in the account under the Exophones section .

ExoPhone

EXOPHONE	TYPE
095-138-86363 ❓ Pin: 7022-1678-17	Not set
080-471-12327	Landline
sip-:08-040408080	VoIP

SIP Exophones discussed later in the section refers to Exophones of Voip type as shown above.

NOTE :- SIP and VOIP are used interchangeably in the document

3. Glossary

Terminology	Description
App	Mobile application
Application Backend	Customer backend service for the application
Client	User / Subscriber / Client signing up to use the mobile app
Customer	Exotel's customer licensing the SDK
Voip	Voice over IP

4. Android SDK Integration

4.1. Getting Started

4.1.1. Software Package
The *ExotelVoice* SDK software package includes
- Android AAR file
- Integration Guide
- Sample application for reference
- JavaDoc for API reference

4.1.2. Add exotel Voice SDK Library to Project
Method I:
- Open your Android application code in Android Studio
- Choose File -> New -> New Module -> Import exotel-voice-release.aar Package
- Provide the path to the downloaded AAR file
- Add the SDK as a dependency for the app
  a. Go to File -> Project structure -> Dependencies
  b. Add Exotel Voice SDK as dependency to your application

Method II:
- Copy the AAR file to the libs folder and edit the build.gradle file to include

```
implementation 'com.google.code.gson:gson:[2.8.5]'
implementation 'dnsjava:dnsjava:[2.1.9,)'
implementation 'com.squareup.okhttp3:okhttp:[3.14.2,)'
implementation 'com.squareup.okhttp3:logging-interceptor:[3.14.2,)'
implementation project(path: ':exotel-voice-sdk')
```

- Below contents part of settings.cradle file

```
include ':app', ':exotel-voice-release'
implementation fileTree(include: ['*.jar'], dir: 'libs')
```

4.1.3. Target Platform
- Supported Android Version: 5.0+
- Supported Android ABIs: armeabi-v7a, arm64-v8a, x86_64 and x86

4.1.4. Permissions

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.BROADCAST_STICKY" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
```

4.1.5. Proguard Rules

The proguard rules for *ExotelVoice* are included in the AAR file. These rules will be auto included when the proguard is enabled for your application build.

4.1.6. Android Service

The application needs to integrate *ExotelVoice* in a service. This should be run as a foreground service when a call is in progress so that the application is not killed when it moves to the background. When the call is over, service should be moved to the background. Refer to *VoiceAppService* in the sample application.

4.2. Initialize Library

ExotelVoice Interface Class provides an entry point to initialize the voice library and set it up for handling calls.

```
// Get Exotel Voice Client
ExotelVoiceClient exotelVoiceClient =
ExotelVoiceClientSDK.getExotelVoiceClient();

//set the event listener for sdk logs.
exotelVoiceClient.setEventListener(this);

// Set listener to handle events from voice client
exotelVoiceClient.setEventListener(new ExotelVoiceClientListener() {

    // Initialization success handler
    void onInitializationSuccess() { ... }

    // Initialization failure handler
    void onInitializationFailure(ExotelVoipError exotelVoipError) { ... }

    // Logs reported by the voice library
    void onLog(LogLevel logLevel, String Tag, String Message) { ... }

    // Log upload success handler
    void onUploadLogSuccess() { ... }

    // Log upload failure handler
    void onUploadLogFailure(ExotelVoipError exotelVoipError) { ... }

    // Authentication failure handler
```

```
        void onAuthenticationFailure(ExotelVoipError exotelVoipError) { ... }
});
```

SDK initialization requires a *subscriberToken* generated by the exotel platform. Workflow for this token generation and management is described in section *Authentication and Authorization*.

```
// Request token from application backend (example)
String subscriberToken = getAccessToken();
```

The *subscriberToken* is provided to *ExotelVoice* during initialization.

```
// Initialize client library
android.content.Context context = this.getApplicationContext();
exotelVoiceClient.initialize(
        context,              // Android application context
        hostname,             // Host URL for the Exotel Voice Platform
        subscriberName,       // Username of subscriber used to generate token
        displayName,          // Display name of the subscriber
        accountSid,           // Exotel Account SID
        subscriberToken,      // Subscriber token fetched from exotel platform
        );
```

Client library notifies *onInitializationSuccess()* on success and *onInitializationFailure()* on failure. On expiry of *subscriberToken*, the library will post onAuthenticationFailure() at which application should request new *subscriberToken* from application backend and program in the *initialize()* API. *SubscriberName* is a 16 byte alphanumeric string.

Below are the sample methods to get the subsriberToken, Customers can implement this in their own method.
Use the below HTTP API's to get the subscriber token and covert subscriber token to jobject and pass it to `exotelVoiceClient.initialize();`

Note:
- Copying manually generated subscriber token has issues. Best way is to make API calls in source code and pass them by converting to jobject to the `exotelVoiceClient.initialize();`
- Pass the **proper device ID** to the login api to avoid invalid refresh token error.
- build.cradle should be proper. Complete working build.cradle file is attached in the link https://github.com/exotel/exotel-voip-sdk-android/blob/main/build.gradle

Generate access token:
Request URL: http://bellatrix.apac-sg.exotel.in/v1/login
HTTP Method: POST
Header:
content-type: application/json
Request Body:

```
{
"user_name":"9980921403",
"password":"exotel321",
"account_sid":"exotel675",
"device_id":"fd09d9a10e028b45",
"platform":"android"
}
```

Response:

```
{
"request_id":"0b47aa6b221b4583a82d24fb68ffbacb",
"method":"POST",
"http_code":200,
"account_sid":"exotel675",
"host_name":"https://miles.apac-sg.exotel.in/v2",
"exophone":"08045687944",
"subscriber_name":"9980921403",
"subscriber_token":
{
"refresh_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJFeG90ZWwwiLCJ
zdWIiOiI5OTgwOTIxNDAzIiwiaWF0IjoiMTY1NTgwNDkwNyIsImV4cCI6IjE2NTU4OTEzM
DciLCJjbGllbnRfaWQiOiJleG90ZWww2NzUiLCJkZXZpY2VfaWQiOiJmZDA5ZDlhMTBlMD
I4YjQ1In0.KlMtyXI5mz86Y4PsRWcgJC2cv_s03LdZ4eSFLKhE33k",
"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJFeG90ZWwwiLCJ
zdWIiOiI5OTgwOTIxNDAzIiwiaWF0IjoiMTY1NTgwNDkwNyIsImV4cCI6IjE2NTU4MDY3
MDciLCJjbGllbnRfaWQiOiJleG90ZWww2NzUiLCJkZXZpY2VfaWQiOiJmZDA5ZDlhMTBl
MDI4YjQ1Iiwic2NvcGUiOiJ2b2ljZSJ9.hgbXmwlkvBRw6rR4FVjXduaEQ3abNoGhuTohAP
ciE6s"
}
}
```

Renew access token:

Request URL:

HTTP Method: POST
Header:
content-type: application/json
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJFeG90ZWwwiLCJzdWIiOiI5OTgw
OTIxNDAzIiwiaWF0IjoiMTY1NTgwNDkwNyIsImV4cCI6IjE2NTU4OTEzMDciLCJjbGllbnR
faWQiOiJleG90ZWww2NzUiLCJkZXZpY2VfaWQiOiJmZDA5ZDlhMTBlMDI4YjQ1In0.KlMt
yXI5mz86Y4PsRWcgJC2cv_s03LdZ4eSFLKhE33k
Request body:
```
{
"platform": "android",
"device_id": "fd09d9a10e028b45",
```

"device_check": "device_check"
}
Response:
{
"request_id":"e61639aaa1e848a2bda0637398b1ccb7",
"method":"POST",
"http_code":200,
"Response":
{
"code":200,
"error_data":null,
"status":"success",
"data":
{
"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJFeG90ZWwwiLCJ
zdWIiOiI5OTgwOTIxNDAzIiwiaWF0IjoiMTY1NTgwNzIxMiIsImV4cCI6IjE2NTU4MDkwMT
IiLCJjbGllbnRfaWQiOiJleG90ZWw2NzUiLCJkZXZpY2VfaWQiOiJmZDA5ZDlhMTBlMDI4
YjQ1Iiwic2NvcGUiOiJ2b2ljZSJ9.ZjxGLdjowuvDlMv2A2f45LMjBAz-usQOak_1Vg332ug"
}
}
}

Profile details:
Request URL:
http://miles.apac-sg.exotel.in/v2/accounts/exotel675/subscribers/9900365989/profile
HTTP Method: GET
Header:
content-type: application/json
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJFeG90ZWwwiLCJzdWIiOiI5OTgw
OTIxNDAzIiwiaWF0IjoiMTY1NTgwNzIxMiIsImV4cCI6IjE2NTU4MDkwMTIiLCJjbGllbnRfa
WQiOiJleG90ZWw2NzUiLCJkZXZpY2VfaWQiOiJmZDA5ZDlhMTBlMDI4YjQ1Iiwic2Nvc
GUiOiJ2b2ljZSJ9.ZjxGLdjowuvDlMv2A2f45LMjBAz-usQOak_1Vg332ug
Request Body: Not required
Response:
{
"request_id":"c4a44b9e45ec4778b00c187709a059c1",
"method":"GET",
"http_code":200,
"subscriber_profile":
{
"sip_uri":"sip:9900365989@exotel.com",
"sip_password":"dummy",
"sip_server":"proxy.apac-sg.exotel.in",
"sip_port":"10001",
"sip_transport":"TLS",
"log_destination_base_url":"https://miles.apac-sg.exotel.in/v2",

"client_events_base_url":"https://miles.apac-sg.exotel.in/v2",
"media_timeout":30,
"file_log_level":"DEBUG",
"console_log_level":"ERROR",
"event_thread_interval":20,
"incoming_call_timeout":30,
"log_rotate_days":30,
"provider_license_key":"KQEgqp1Kifew0QEJ4p6U6yovtJkgsBhksUSb1iynO0wJpPtwSy
5deQkwWZMrAACHeV5wFWEayQAwn5OsWBJpfg2uLq98e3xh3HOHu3cU6akYRycQE
zlEijl6TawVFm99otPcSkTB5+ERRTO6UfeRgJHrMNiQI5NFLI9FQHRYiOyFQHTFgqkJFh
uiLrtwvBFDvCFoq7QWDC8b4lfHFKLfqRNlgxft3YFvYt15xCHB/D9SjKu9cUEgTJylmFRv
srklnKJBVxqlM8jfWyB2UUooWBSTSf2DPMG75GstsQWYo8U5ISogYJ+MvLKPh4nn/NE
yVUdEMxEw9gkn+sjSKRDJBw=="
}
}

4.3. Managing Calls

After successful initialization, the library is ready to handle calls. Dialing-out calls or receiving incoming calls is managed through *CallController* Interface.

```
// Get Call Controller interface
CallController callController = exotelVoiceClient.getCallController();
```

Call progress events are notified to the application through the *CallListener* Interface attached to the *CallController* object.

```
// Attach call listener
callController.setCallListener(new CallListener() {

    // Handler for incoming call alert
    void onIncomingCall(Call call) { ... }

    // Handler for outgoing call initiated event
    void onCallInitiated(Call call) { ... }

    // Handler for call ringing event for outgoing call
    void onCallRinging(Call call) { ... }

    // Handler for call connected event
    void onCallEstablished(Call call) { ... }

    // Handler for call termination event
    void onCallEnded(Call call) { ... }

    // Handler for missed call alert
    void onMissedCall(String remoteId, Date time) { ... }
});
```

ExotelVoice library always routes the call via SIP Exophone configured for your account. SIP Exophones are different from the PSTN / Mobile Exophones. They can be identified using sip: prefix. Contact *exotel support* to enable SIP Exophone in your account.

Application must program SIP Exophone number in the *dial()* API for outgoing calls.

```
// Dial out call to remoteId
Call call = callController.dial(<sip exophone>);
```

Similar to the number masking workflow, the application backend must maintain the call context between caller and callee. On receiving the call at SIP Exophone, the exotel platform will request the callee details from the application backend to bridge the call. Refer section *Dial Whom Endpoint* for details.

Each call object provides a Call Interface to manage the call lifecycle and perform operations like answer incoming calls, hangup calls, mute, unmute, and get call statistics.

```
// Mute
call.mute();

// Terminate call
call.hangup();
```

4.3.1.   Make Outgoing Call

To make outgoing calls, the *CallController* interface must be created and a listener interface is attached as mentioned in section *Managing Calls*. Provide sip exophone number in *dial()* API of *CallController* Interface to make outgoing calls.

```
// Get Call Controller interface
CallController callController = exotelVoiceClient.getCallController();

// Attache call progress listener
callController.setCallListener(new CallListener() { ... }

// Dial out call to remoteId
Call call = callController.dial(<sip exophone>);
```

Once initiated, call progress events can be monitored through *CallListener* Interface and call control operations can be performed using *Call* Interface. After dialing, the application can use *hangup()* API on the call object to cancel / terminate outgoing calls.

```
// Cancel / Terminate call
call.hangup();
```

4.3.2.   Receive Incoming Call

The Mobile application receives incoming call data in Push Notification sent by your application backend. Refer section *Push Notification Endpoint* for details.
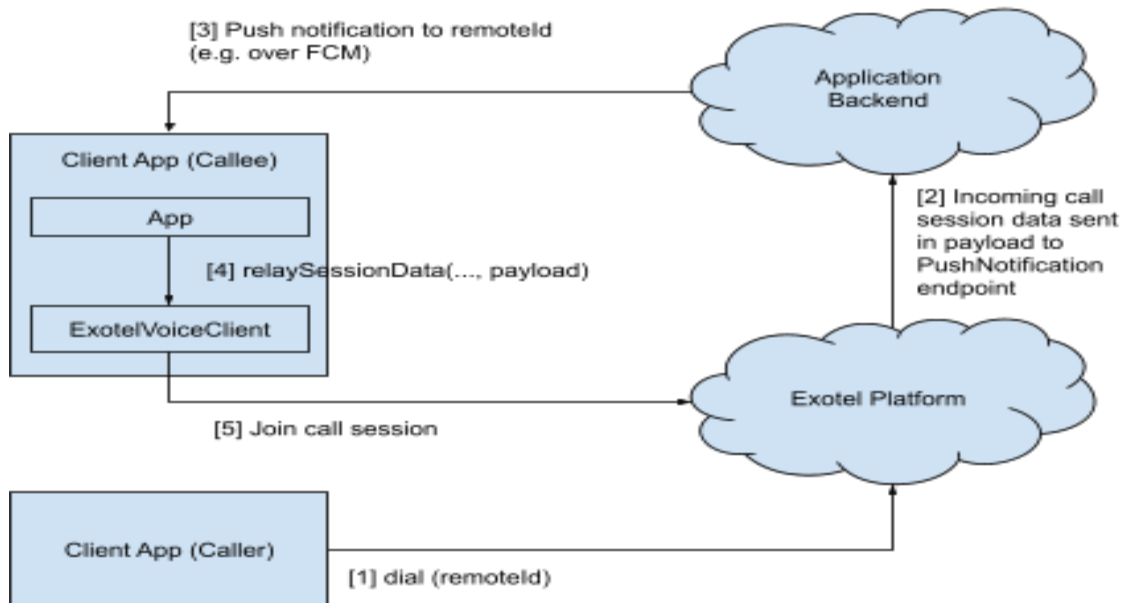
To handle the incoming call, the *ExotelVoice* library should be in an initialized state. Refer to section *Initialize Library*. After initialization, the CallController Interface should be created and the listener be attached as mentioned in section *Managing Calls*.

```
// Get Call Controller interface
CallController callController = exotelVoiceClient.getCallController();

// Attache call progress listener
callController.setCallListener(new CallListener() { ... }
```

The sample application uses Firebase Cloud Messaging to push call data to the device. Once received at the mobile app, it is sent to *ExotleVoiceClient* through *relaySessionData* API.

```
public class VoiceAppFirebaseService extends FirebaseMessagingService{
    public void onMessageReceived(RemoteMessage remoteMessage) {

        Map<String,String> callData = remoteMessage.getData();
        Map<String,String> pushNotificationData = new HashMap<>();

        pushNotificationData.put("payload", callData.get("payload"));
        pushNotificationData.put(
                    "payloadVersion", callData.get("payloadVersion"));

        exotelVoiceClient.relaySessionData(pushNotificationData);
    }
}
```

The voice library will process and validate the data and send an *onIncomingCall()* event to the application with the call object. The application can call the *answer()* API to accept the incoming call.

```
// Accept call
call.answer();
```

The application can call *hangup()* API on the call object to decline an incoming call.

```
// Decline / Terminate call
call.hangup();
```

4.4. Audio Management

ExotelVoice can detect the change in audio output selection. Default audio output mode is the earpiece. When a wired headset is plugged in or a Bluetooth headset is paired with a phone, it automatically routes audio over a new route. Speaker phone mode can be enabled using *Call* object APIs.

```
// Enable speaker mode
call.enableSpeaker();

// Disable speaker mode
call.disableSpeaker();
```

Local mic can be muted and unmuted during in-call using *Call* object APIs.

```
// Enable speaker mode
call.mute();

// Disable speaker mode
call.unmute();
```

4.5. Call Statistics

Application can query the call quality statistics periodically during the call by using *getStatistics()* API of the *Call* object. It provides info about codec and the call QoS parameters like packet loss, jitter and round trip delay.

4.6. Call Details

Application can query the call details record of the call using *getCallDetails()* API of the *Call* interface. The CDR provides info on callId, call state, call duration, call end reason etc. This API can be queried only for the latest call since SDK maintains only a single call context.

4.7. Handling of PSTN calls

ExotelVoice has following behavior during PSTN calls:
- When a PSTN call is in progress, no outgoing calls are allowed by the voice client.
- When a PSTN call is in progress, any incoming VoIP call is automatically declined by the voice library and reported as a missed call to the application.
- When a VoIP call is in progress and a PSTN call lands on the phone, then the VoIP call continues if the user declines or does not respond to the PSTN call. If the user accepts the PSTN call, then the VoIP call is automatically terminated by the voice client.

4.8. Reporting Problems

The voice client library logs are stored in the application internal storage. Any issue along with the logs can be reported by app to Exotel using *uploadLogs()* API of *ExotelVoice* Interface.

```
// Upload logs with description from startDate and endDate
exotelVoiceClient.uploadLogs(startDate, endDate, description);
```

The API triggers *onUploadLogSuccess()* or *onUpLoadLogFailure()* callback based on the success or failure of the operation.

4.9. Reporting Call Quality Feedback

Call quality can be queried from the user and reported to the exotel platform at the end of the call.

```
// Upload logs with description from startDate and endDate
int rating = 3
CallIssue issue = BACKGROUND_NOISE
call.postFeedback(rating,issue);
```

The rating is a quality score that can take values 1 to 5.

```
5 - Excellent quality. No issues.
4 - Good quality. Negligible issues.
3 - Average quality. Minor audio noise.
2 - Bad quality. Frequent choppy audio or high audio delay.
1 - Terrible. Unable to communicate. Call drop, No-audio or one-way audio.
```

The call issue is a descriptive explanation of the issue.

NO_ISSUE	No issues observed
BACKGROUND_NOISE	Low audio clarity due to noisy audio
CHOPPY_AUDIO	Frequent breaks in audio or garbling in audio

HIGH_LATENCY	Significant delay in audio
NO_AUDIO	No audio received from far user
ECHO	Echo during the call

5. Platform Integration

5.1. Customer API Endpoints

Exotel provides full control to customers to implement call routing business logic. For this, customers need to host the following HTTP endpoints to handle callbacks from Exotel platform.

5.1.1. Dial Whom Endpoint

Host a HTTP endpoint which will be queried by the Exotel platform to get to the destination user.

Method: GET
Request URL (Example): https://company.com/v1/accounts/exotel/dialtonumber
Note: This URL needs to be provided to Exotel or configured in the connect applet.

Request Body:
● CallSid - unique call identifier
● CallFrom - caller username
● CallTo - exophone

Expected Response:
On success, the API should return with response code 200 OK. The response body should be a string of type *sip:<remoteId>*. "sip:" tag is needed to hint the exotel platform to connect calls over voip. At present, SIP and PSTN intermixing is not supported. Any other response is treated as failure. remoteId is the username with which the call destination subscriber was registered with Exotel platform.

Example:
Request

```
GET
/v1/accounts/exotelip2ipcalling1/dialtonumber?CallSid=743ddcf5a0552050bc37b6d0
ff9613bn&CallFrom=sip:Alice&CallTo=sip:08040408080&CallStatus=ringing&Directio
n=incoming&Created=Sat,+23+Nov+2019+22:00:37&DialCallDuration=0&StartTime=2019
-11-23+22:00:37&EndTime=1970-01-01+05:30:00&CallType=call-attempt&DialWhomNumb
er=&flow_id=249196&tenant_id=113828&From=sip:Alice&To=sip:08040408080&CurrentT
ime=2019-11-23+22:00:37
```

Response

```
HTTP/1.1 200 OK
{
    sip:Bob
}
```

5.1.2.   Push Notification Endpoint

Exotel implements registration-less dialing where a user need not periodically register with SIP registrar for receiving incoming calls. Instead the call details are pushed to the device as push notification using which call can be established. This method is beneficial as calls will reach the user even when the app is not in foreground or swipe killed. It saves battery since it does not need to keep persistent voip connection when idle.

Exotel will provide call data to this endpoint that needs to be pushed to the client device from your application backend.

Method: POST
Request URL (Example):
https://<your_api_key>:<your_api_token>@company.com/v1/accounts/exotel/pushtoclient
Note: This URL needs to be configured in Exotel platform

Request Body:
● subscriberName - Name with which subscriber registered with Exotel platform
● payload - call data which should be passed to the client SDK
● payloadVersion - version of payload scheme

Expected Response:
On success, the API should return with response code 200 OK. Any other response is treated as failure.

5.2.   Exotel API Endpoints

5.2.1.   Subscriber Management

Customers can manage their subscribers in the exotel platform using the APIs listed *"Subscriber-Management-API"* document.

For clients to be able to use voip calling features they need to be added as subscribers under your account. There are two ways in which your client registration with Exotel account happens,
a. Pre-provisioning
   The customer pre-configures the client accounts even before the app is installed by the client. Refer to the *"Subscriber-Management-API"* document for details on creating client accounts.

b. Dynamic provisioning
   A client account is created after the app is installed by the user and signs-up with the application backend. Refer to *Authentication and Authorization* for workflow details.

Once client provisioning happens, clients can be managed using Subscriber Management APIs.

6. Authentication and Authorization

Access to the exotel platform by *ExotelVoice* is authenticated using a set of bearer tokens - *subscriber_token*. The Application backend must obtain these tokens from the exotel platform and provide them to the client on request. Refer to the *"Subscriber-Management-API"* document for details.
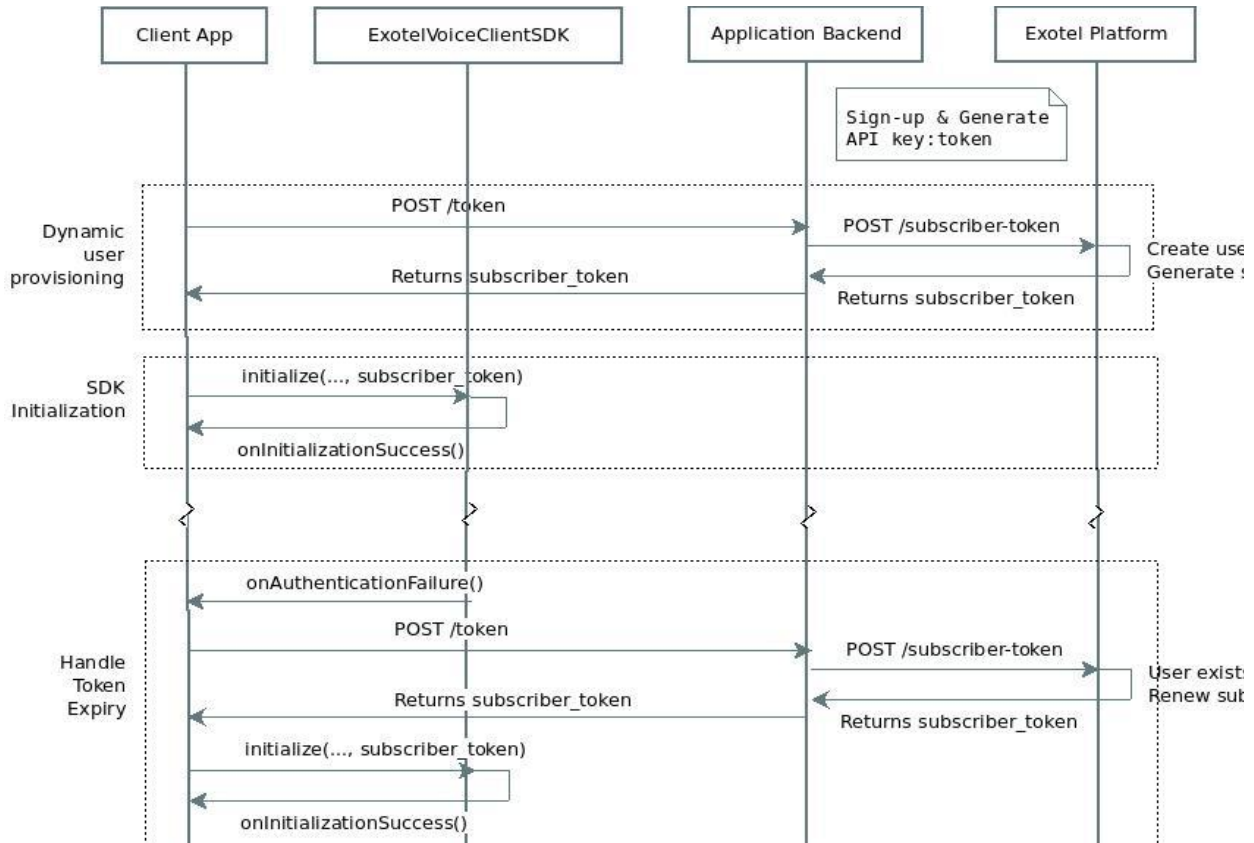
On receiving the *onAuthenticationFailure* event, the application should request a new *subscriber_token* from its backend and reinitialize the SDK as shown in section *Initialize Library*. Contact *exotel support* if you get *onAuthenticationFailure* even after token renewal.

onAuthenticationError() event provides following error types:
- AUTHENTICATION_INVALID_TOKEN - token parameters are invalid
- AUTHENTICATION_EXPIRED_TOKEN - token expired

7. Reference Documents

- Android JavaDoc for the *ExotelVoice* library API
- *Subscriber-Management-API* document for details on API to manage subscriber

8. Support Contact

Please write to hello@exotel.in for any support required with integration.

9. Troubleshooting Guide

9.1. Outgoing Call

a. I am not getting any requests on the "Dial Whom" endpoint for outgoing calls?

i. Check if *"Dial Whom"* URL was configured in *[Connect Applet](#)* and reachable.
ii. Check if the call is listed in your account dashboard. If not, then
- Check if the phone has internet connectivity.
- Check if the SIP Exophone number was set in *CallController::dial()* API.
- Check if *CallListener::onCallFailed()* event was received with error type CONGESTION_ERROR, which means rate limit quota was exceeded.
- Check if *CallListener::onCallInitiated()* event was received from the SDK.
- Check if *Call::getCallDetails()* returns non-null *sessionId* field
- Check if
iii. Report the issue to *[exotel support](#)* with *sessionId* (from App) and *Call-Reference-Id* (if available, from account dashboard)

b. I am hearing a ringing tone but the callee has not received the call?

Ringing tone implies that the call has reached the exotel platform.

9.2. Incoming Call

a. My calls are not landing on the callee app?

i. Check if the call is listed in your account dashboard and dialout happened to the remote subscriber. If not then, refer to the troubleshooting guide for outgoing calls.
ii. Check if the application received push notification with call payload from your application backend. If not, then
- Check the device OEM. If Xiaomi phones, refer to section *[Problems with push notifications on Xiaomi devices](#)*.
- Check if *[Push Notification URL](#)* is reachable and configured in the exotel platform.
- Check if Push Notification URL got the call payload from exotel platform.
- Check if the call payload was sent to the target device from your backend.
iii. If the call notification is received in the app, then
- Check if the library is initialized using *ExotelVoiceClient::isInitialized()*

- Check if payload received in push notification was programmed in the library through *ExotelVoiceClient::relaySessionData()*

iv. Report the issue to *exotel support* with *Call-Reference-Id* (from account dashboard)

b. Problems with push notifications on Xiaomi devices

i. Your application needs to have the following additional permissions in the MIUI settings page*. Open settings (by tapping the cog icon in the top right corner), select "Manage Apps", select "your application"
- Enable "Autostart"
- Select "Other permissions"
  - Turn on "Show on Lock Screen"
  - Turn on "Display pop-up windows while running in the background"
- Select "Battery Saver" and set "No restrictions"

* The steps here refer to MIUI Global 11.0.2 on Android 7.1. Depending on device and software version, designation and location of menu items may differ.

9.3. Authentication

a. Requesting subscriber-token from exotel platform giving 4xx response?

Response Code	Troubleshooting
400	Malformed packet. Check subscriber_name format. It should be an alphanumeric string of size less than 16 characters.
401	It requires basic authentication using API key:token
403	VoIP calling is not enabled in your account. Contact exotel support.

b. Why is SDK reporting *onAuthenticationFailure* response for a subscriber token received from exotel platform?

i. Check the *ErrorType* in response. If it is,

● AUTHENTICATION_INVALID_TOKEN	Invalid Token. Initialize new token
● AUTHENTICATION_EXPIRED_TOKEN	Token expired. Initialize new token

ii. Contact *exotel support* if the issue persists.