Client Samples

Android SDK Sample

9/13/2025

Contents

- 1 Android SDK Sample
  - 1.1 Getting Started
  - 1.2 Using Settings
  - 1.3 ChatHandler
  - 1.4 Firebase Cloud Messaging
  - 1.5 Disclaimer

Android SDK Sample

Introduced in 8.5.204.00

Important

The SDK described on this page is a sample, not a Genesys product. The related-code is meant to be used with the Android Sample for demonstration purposes only. See the Disclaimer section for further details.

The Android SDK used in this sample is available in jcenter/bintray and in Maven, and includes the Javadoc JAR file that you can import in Android Studio. Once imported, the JAR file also provides automatic completion and documentation while using the Android Sample SDK.

Getting Started

Use your favorite editor to edit the `app/build.gradle` file.

1. Add the GMS SDK with the correct version to the dependencies sections:

```
repositories {
        jcenter()
}
dependencies {
    ...
        implementation 'com.genesys.gms.android:sdk:<version_from_maven_repo>'
    ...
}
```

For example, if you download your SDK from the revision 241 use `8.5.201.00.rev.241` for the version.

2. Ensure that the Android `minSdkVersion` option is not less than 19 and that the Android `compileSdkVersion` option matches the latest version available (here, 27).

```
android {
    compileSdkVersion 27
    defaultConfig {
        ...
            minSdkVersion 19
        ...
    }
}
```

Structure

The Android SDK is divided into two parts:

- `Settings`: Handles all the settings that the API needs to connect with GMS and Callback.
- `Chat`: Handles all the actions required during a Chat session.

See the sections below for further details on using these APIs.

Important

The Javadoc is included in your JAR file.

Using Settings

First of all, to use the Settings part of the SDK, you need to implement the `SettingsActivity` class and create a `SettingsManager` instance. This instance contains and handles all the settings that you need for each subsequent actions.

For example:

```
public class SettingsActivity extends Activity implements SettingsHandler {

    [...]
    SettingsManager settingsManager;

    public void aMethod() {

        //Initialize settingsManager
        //SettingsManager(Context, SettingsActivity)
        settingsManager = SettingsManager.createInstance(this, this);
    }

    @Override
    public void onError(Exception exception) {

    }

    @Override
    public void dial(DialogResponse dialog) {

    }

    @Override
    void availableTimeslots(Map<String, String> var1){

    }

}
```

To retrieve the `SettingsManager` instance after the initialization, use the `getInstance` method.

```
SettingsManager settingsMgr = SettingsManager.getInstance();
```

In this handler, you can implement the `dial` method to process the dialog response. The example below provides an example of implementation:

```
@Override
public void dial(DialogResponse dialog) {
    switch (dialog.getAction()) {
        case DIAL:
            Log.d("Label", dialog.getLabel());
            callbackApi.makeCall(Uri.parse(dialog.getTelUrl()));
            break;
        case MENU:
            if (dialog.getContent() == null || dialog.getContent().size() == 0) {
                // It's empty! No menu to show...
                break;
            }
            DialogResponse.DialogGroup group = dialog.getContent().get(0);
            String groupName = group.getGroupName();
            final List<DialogResponse.GroupContent> groupContents = group.getGroupContent();
            if (groupContents == null || groupContents.size() == 0) {
                // It's empty! There are no options...
                break;
            }
            String[] menuItems = new String[groupContents.size()];
            for (int i = 0; i < groupContents.size(); i++) {
                menuItems[i] = groupContents.get(i).getLabel();
            }
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setTitle(dialog.getLabel() + "\n" + groupName)
                    .setItems(menuItems, new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int which) {
                            String url = groupContents.get(which).getUserActionUrl();
                            settingsManager.startDialog(url);
                        }
                    });
            builder.create().show();
            break;
        case CONFIRM:
            String text = dialog.getText();
            Toast.makeText(this, text, Toast.LENGTH_LONG).show();
            break;
        default:
            break;
    }
}
```

Ensure to save all the settings by using the `saveDatas` method:

```
settingsManager.saveDatas(settings);
```

The `settings` object stores all the useful settings, like the name and telephone number of the user and all server settings such as the address, the port, and so on. The `saveDatas` methods allows to save all the settings and get them later or after closing the application. For example, after an `onResume` event in an activity, retrieve the saved data by calling `SettingsManager.getDatas()`:

```
settings = SettingsManager.getDatas();
```

When you start a call or a chat session as detailed in other sections, ensure to configure all the settings and then use the `connect` method like in this snippet:

```
public void sendData () {
    Map<String, String> params;

    settingsManager.saveDatas(getInformations());
    ScenarioType scenario =
ScenarioType.fromString(scenarioSpinner.getSelectedItem().toString());

    params = callbackApi.getScenarioParams(getInformations(), scenario);

    if (scenario == ScenarioType.CHAT_V2) {
        Intent intent = new Intent(this, ChatActivity.class);
        startActivity(intent);
    } else {
        settingsManager.connect(serviceNameTextView.getText().toString(), params);
    }

}
```

Important

Remember that if you want to start a scheduled call, you must set the `dateTime` attribute in the `settings` instance as an ISO 8601 date.

ChatHandler

To start a chat session, create the UI and open your own activity. To create your `ChatActivity` class, extend the `Activity` class and implement the `ChatHandler` interface.

```
public class ChatActivity extends Activity implements ChatHandler {
    [...]
    ChatClient chatClient;

    public void aMethod() {
        //ChatClient(Context, ChatHandler)
        chatClient = new ChatClient(context, this);
    }

    @Override
    public void onMessage(ChatV2Response response){
        [...]
    }

    //The url of the downloaded file
    @Override
    public void onFileDownloaded(String url){
        [...]
    }


    @Override
    public void onFileUploaded(Uri uri){
        [...]
    }


    @Override
    public void onError(Exception e){
        [...]
    }

}
```

First create an instance of `ChatClient`, like in the `aMethod` example above. After initializing the API, you can start a Chat session by calling the `connect` method.

```
chatClient.connect();
```

The `connect` method automatically starts the chat session and call the “onMessage” or “onError” when it is needed. You can send chat start and stop typing events by calling `startTyping` and `stopTyping` methods, as shown below:

```
chatClient.startTyping()
chatClient.stopTyping()
```

To send a message, simply use the `sendMessage` method.

```
chatClient.sendMessage(message)
```

You can also download files through the `downloadFile` method.

```
chatClient.downloadFile(messageId)
```

To download a file from the chat session, use the messageID that is provided in the response of the “onMessage” event. The `downloadFile` method returns the URI of the given file where the user can start the download.

To end the chat session, call `disconnect`:

```
chatClient.disconnect()
```

Firebase Cloud Messaging

Refer to the instructions detailed in the Android Sample’s Readme to implement FCM. Once all the firebase instructions are completed (Connect app, add FCM, access the registration token, handle message), you can receive messages in the `onMessageReceived()` method. Process the message in your handler, for example, as follows:

```
@Override
public void onMessageReceived(RemoteMessage remoteMessage) {
    // Check if message contains a data payload.
    if (remoteMessage.getData().size() > 0) {
        SettingsManager settingsManager = SettingsManager.getInstance();
        settingsManager.handleMessage(remoteMessage.getData().get("message"));
    }
}
```

Disclaimer