Android—Clover Go SDK quick start guide

United States

This quick start guide explains how to take a payment using the Android Clover Payment software development kit (SDK) and your Clover Go reader.

Prerequisites

- See Overview of the Clover platform.
- Create a global developer account with a default test merchant account.
- Create additional test merchants, if needed.
- Order a Clover Go reader Developer Kit (Dev Kit) and set it up.
- Install Android Studio Chipmunk | 2021.2.1 Patch 2 or higher.
- Use an Android-powered device (Android Oreo/8.0+).
- Charge Clover Go reader—Device battery charging requirement. Several operations on your Clover Go reader require at least 30% battery. Charge your device before you configure your iOS project using the instructions in this guide.

OAuth authentication

Clover uses OAuth to authenticate your app's users to the Clover servers. Use the steps in this topic to create a Clover app and install it on your test merchant. The Clover app has an associated App ID and App Secret that the Android app uses to obtain the permissions needed to perform OAuth. See Authenticate with OAuth for more information.

Steps

1. Create your test app in sandbox

1. Log in to your Clover global developer account. The Global Developer Dashboard appears.

2. Click **Create App** and follow the instructions.

3. Use the following settings as a baseline for your test app.

Field	Description
App Type	In the REST Clients section, click **Web**.
Requested Permissions	Select checkboxes based on required permissions. See Set app permissions.
REST Configuration	
- Site URL	Enter the domain and URL (link) that you control for OAuth redirection upon completion.
- Default OAuth Response	Select **Code**. Code and Token responses are both supported; however, only Code responses are used in production because they provide an extra level of security.
Ecommerce	Integrations Enabled: API

4. Note the App ID and App Secret to use later when you configure the Android app.

2. Create your Android app and import the SDK

1. In Android Studio, create a new Android Project.

Field	Description
Activity	Leave blank.
Name	Go SDK Intro
Language	Kotlin
Minimum SDK	API 26: Android 8.0 (Oreo)
Package name and Save location	As per your preference.

2. Add the dependency directly to the module's dependencies in the `build.gradle` file:

Example

```
implementation ("com.clover.sdk:go-sdk:latest.release")
```

3. Configure the SDK

1. Create a file with the name `GoSDKCreator` in your Android project.
2. Copy the following code and paste it into the `GoSDKCreator` file.

Kotlin sample Java sample

```
object GoSDKCreator {

    // App ApiKey and Secret
    private const val API_KEY = YOUR_API_KEY
    private const val API_SECRET = YOUR_API_SECRET
    private fun buildConfig(context: Context) = GoSdkConfiguration.Builder(
        context = context,
        appId = BuildConfig.APPLICATION_ID,
        appVersion = "1.0.0",
        apiKey = API_KEY,
        apiSecret = API_SECRET,
        oAuthFlowAppSecret = YOUR_OAUTH_FLOW_APP_SECRET,
        oAuthFlowRedirectURI = YOUR_OAUTH_FLOW_REDIRECT_URI,
        oAuthFlowAppID = YOUR_OAUTH_FLOW_APP_ID,
        environment = GoSdkConfiguration.Environment.SANDBOX,//Available environments are
        reconnectLastConnectedReader = true
    )
    .build()

    private lateinit var goSdkInstance: GoSdk

    @JvmStatic
    fun get(context: Context): GoSdk {
        if (!GoSDKCreator::goSdkInstance.isInitialized) {
            goSdkInstance = GoSdkCreator.create(buildConfig(context))
        }
        return goSdkInstance
    }
}
```

2. Enter the following configuration in your file:

Property	Description
Environment	Environment you want to connect to on Clover servers. Start with **Sandbox** to build and test your app.
OAuthFlowRedirectURI	Link `http://<yourdomain>/OAuthResponse` where your domain is located. When the OAuth flow starts, this URI is passed to Clover login servers. After the user authenticates, this is the URI that Clover servers call to link back to your app. This value is validated against the site URL you configured on Clover servers before Clover servers call it.
OAuthFlowAppID	App ID of the Clover app you created in sandbox. See step 1.
OAuthFlowAppSecret	App Secret of the Clover app you created in sandbox. See step 1.
APIKey	Ask your Developer Relations Representative for this value.
APISecret	Ask your Developer Relations Representative for this value.

4. Initialize the SDK

1. In your activity or fragment, add this line:

   Kotlin sample Java sample

   ```
   private val goSdk: GoSdk by lazy { GoSDKCreator.get(requireContext()) }
   ```

2. Run your app on an Android-powered device to go to the Clover login window.
3. Enter your credentials in the login window. On successful login, you are redirected to your app.

5. Scan for devices and connect

1. Open a Bluetooth scan, receive a list of devices found, and connect to the first found device.
    - If the device is running Android S (Snow Cone, Android 12) or higher, request BLUETOOTH_CONNECT and BLUETOOTH_SCAN permissions.
    - If the device is not running Android S or higher, determine what Android version the device is running and request ACCESS_FINE_LOCATION permissions using runtime permissions. See the Android documentation for more information.
2. Inside a coroutine scope, add the lines:

Kotlin sample Java sample

```
goSdk.scanForReaders().catch {
    //handle error
}.collectLatest {
    //adapter.addDevice(it) //In case you have a scan list
    // OR in case you want to connect to a particular device
    if (it.bluetoothName.contains("XXXXXX")) { //XXXXXX in this case, is the las
        goSdk.connect(it)
    }
}
```

6. Take a payment

Clover recommends that you wait to initiate a charge request until the `CardReaderStatus.READY` event appears. The card status appears regardless of the last action.

1. To observe the `CardReaderStatus`:

Kotlin sample Java sample

```
goSdk.observeCardReaderStatus(viewLifecycleOwner, cardReaderStatusCallback)

val cardReaderStatusCallback = object : GoSdkCallback<CardReaderStatus> {
    override fun onError(e: Throwable) {
        //handle error scenario
    }

    override fun onNext(result: CardReaderStatus) {
        // publish it or save in ViewModel?.. anything you want
        when (result) {
            is CardReaderStatus.BatteryPercentChanged -> {}
```

```
            is CardReaderStatus.BleFirmwareUpdateInProgress -> {}
            is CardReaderStatus.CheckingForUpdate -> {}
            is CardReaderStatus.Connected -> {}
            is CardReaderStatus.Connecting -> {}
            is CardReaderStatus.Deleted -> {
                //this is where someone will update UI to disable payment taking UI
            }
            is CardReaderStatus.Disconnected -> {
                //this is where someone will update UI to disable payment taking UI
            }
            is CardReaderStatus.Ready -> {
                //this is where someone will update UI to take a payment
            }
        }
        is CardReaderStatus.ResetInProgress -> {}
        is CardReaderStatus.RkiInProgress -> {}
        is CardReaderStatus.SpFirmwareUpdateInProgress -> {}
        is CardReaderStatus.UpdateComplete -> {}
    }
}
```

2. Inside a coroutine scope, call the charge function in the GoSDK interface.

Kotlin sample Java sample

```
val request = PayRequest(
    final = true, //true for Sales, false for Auth or PreAuth Transactions
    capture = true, //true for Sales, true for Auth, false for PreAuth Transactions
    amount = 100L,
    taxAmount = 0L,
    tipAmount = 0L,
    externalPaymentId = "pay-id", //Can be null
    externalReferenceId = "invoice-num", //can be null
    keyedInCard = null //Card value can be filled in, if you want to take payment w/o card
)

// Clover recommends that you wait to initiate a charge request until the CardReaderStatus

goSdk.chargeCardReader(
    request
).collectLatest {
 //We suggest logging or updating the UI with the transaction status so you follow the tra
 //and to know if the device is waiting for a user action (i.e. if the device is waiting f
 updateUIWithCardReaderState(it)
}

private fun updateUIWithCardReaderState(it: ChargeCardReaderState) {
    updateTransactionLog(
        when (it) {
```

```
            is ChargeCardReaderState.OnPaymentComplete -> "OnPaymentComplete: ${it.respons
            is ChargeCardReaderState.OnPaymentError -> "OnPaymentError: $it"
            is ChargeCardReaderState.OnReaderPaymentProgress -> "OnReaderPaymentProgress:
        }.toString()
    )
}
```

7. Install and run the app

When the app first launches, the Clover login window opens in Chrome.

1. In the Chrome browser window, enter your login details.
2. In the app, approve the required permissions.

The app scans for devices. The app connects to the first found device, and starts a charge.

Kotlin Sample Java sample

```
package com.example.gosdkintro

import android.Manifest
import android.content.DialogInterface
import android.content.pm.PackageManager
import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.core.app.ActivityCompat
import androidx.lifecycle.lifecycleScope
import com.clover.sdk.gosdk.GoSdk
import com.clover.sdk.gosdk.model.PayRequest
import com.clover.sdk.gosdk.payment.domain.model.CardReaderStatus
import kotlinx.coroutines.flow.collectLatest
import kotlinx.coroutines.launch

class MainActivity : AppCompatActivity() {
    private val goSdk: GoSdk by lazy { GoSDKCreator.get(this) }

    private val requiredPermissions = when {
        Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> arrayOf(
            Manifest.permission.BLUETOOTH_CONNECT,
            Manifest.permission.BLUETOOTH_SCAN,
        )
        Build.VERSION.SDK_INT <= Build.VERSION_CODES.P -> arrayOf(
            Manifest.permission.ACCESS_COARSE_LOCATION
```

```
        )
        else -> arrayOf(
            Manifest.permission.ACCESS_FINE_LOCATION
        )
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        ActivityCompat.requestPermissions(
            this,
            requiredPermissions,
            REQUEST_PERMISSION_CODE
        )
```

Updated 5 months ago

← Clover Go SDK quick start guides

iOS—Clover Go SDK quick start guide →