Version 2.3.1:1

Introduction

The goal of this ActiveSDK Documentation is to give any developer with iOS or Android experience the knowledge and resources required to integrate GPayments's 3-D Secure 2 3DS SDK (ActiveSDK) into an existing 3DS Requestor App, which will in turn provide cardholders with a seamless ecommerce authentication experience.

This documentation will introduce ActiveSDK, explain how to integrate ActiveSDK to a merchant app, and provide troubleshooting procedures.

Introduction to ActiveSDK

Welcome to ActiveSDK!

ActiveSDK is a 3-D Secure 2 solution provided by GPayments Pty Ltd.

Mobile commerce has become increasingly prevalent around the world. The value of payments made via mobile devices is expected to reach US$1 trillion in 2019, with mobile wallets expected to become more popular than debit and credit cards by 2020. With so many now using their devices to purchase products and log in to a multitude of online services through mobile applications, the risk of mobile fraud is higher than ever.

ActiveSDK offers native mobile functionality for 3-D Secure 2 authentications. ActiveSDK is developed in accordance with EMVCo's specifications, and includes 2 separate SDKs; one for Android, and one for iOS. ActiveSDK can be integrated with iOS and Android apps to connect to any 3DS Server, and complete 3-D Secure authentications.

ActiveSDK enables rich-data exchange between the mobile device and the ACS for risk-based decision making, allowing for an immediate decision on whether to allow or challenge a transaction. ActiveSDK will also provide for native authentication screens, which will help the 3-D Secure interface seamlessly with a merchant's mobile application, and provide a look and feel consistent with the rest of the in-app purchase process. It has full ActiveServer integration support by default and comes with complete documentation and demo app with sample code to help integrate into your application quickly.

For documentation on ActiveServer, which is GPayments's 3DS Server solution, please visit:

- https://docs.activeserver.cloud/en/

Supported versions

The following table shows the minimum supported version for ActiveSDK:

Platform	Minimum Version
iOS	iOS 9
Android	4.4 (API Level 19)

Note

Windows 10 Mobile is not supported.

Supported EMVCo 3-D Secure Protocol Version Number

ActiveSDK (for both iOS and Android) supports the following EMVCo 3-D Secure 2 protocol version numbers.

Protocol Version Number	Status
2.1.0	Supported
2.2.0	Supported

Concepts to Understand

ActiveSDK is a standard implementation of EMVCo's 3-D Secure 3DS SDK specification. Understanding the concepts of 3-D Secure will make this documentation easier to follow, and would allow app developers to fully utilise the capability of ActiveSDK.

For useful resources on 3DS SDK specifications, please see:

- EMV® 3-D Secure Protocol and Core Functions Specification.pdf
- EMV® 3-D Secure—SDK Specification.pdf
- Definitions

Integration

Integration section includes detailed tutorials on how to integrate ActiveSDK with Android and iOS apps. All 3DS SDK authentication processes and their work flows are covered.

- Integration section

3DS Requestor

To integrate ActiveSDK, the merchant site need to implement a 3DS Requestor in the backend. This section takes you through the process of integrating the Active SDK with our 3DS Requestor demo code. In this demo, the 3DS Server which the 3DS Requestor connected to is our ActiveServer.

- 3DS Requestor section

Integration guide

This section includes detailed tutorials on how to integrate ActiveSDK with Android and iOS apps. All 3DS SDK authentication processes and their work flows are covered.

Security Guidance and Best Practices

Before starting development using the 3DS SDK, it is important to consider the security implications, and how to best protect sensitive data. While the SDK takes care of many security functions, as detailed in the EMVCo 3DS SDK specification, there are other considerations that need to be taken into account.

Communication with the 3DS Server

While communication with the 3DS Server is outside the scope of the 3DS SDK itself, the data transmitted, both generated from the 3DS SDK and otherwise, should be properly secured according to payment system security standards. Annex D of the `EMVCo Protocol and Core Functions Specification` provides guidance that has been summarized below. These standards would be appropriate for communication between the 3DS SDK and 3DS Server:

TLS 1.2 or higher should be used, with key lengths as follows:

- **RSA:** 2048 bits or longer.
- **ECC:** 256 bits or longer.

One of the following cipher suites should be used:

- **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256**
- **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256**

Curve P-256 shall be used and indicated in the cipher suite extension.

Protection of Sensitive Cardholder Data

Much of the data used for authentication is outside the scope of the 3DS SDK, and is transmitted to the 3DS Server outside any authentication parameters. This sensitive data includes anything from the card number itself to cardholder contact details.

Protection includes the above secure transmission, as well as protection on the device itself. This may include either not saving the data locally, or encrypting it properly if stored.

Cleanup the ThreeDS2Service Instance

Make sure to cleanup the SDK after you are finished with 3-D Secure. This can be done using the `cleanup` method like below:

Android iOS

```
ThreeDS2Service.INSTANCE.cleanup(applicationContext);
```

Downloads and Updates

Updates to the 3DS SDK are made available on the GPayments repository. Builds there are updated periodically, as well as when critical security updates are made. When the builds are updated, an email notification will be sent out to SDK clients.

No updates are pushed directly to deployed applications. It would instead be required to obtain the latest builds and integrate them into the application. When updating, care should be taken to ensure the integrity of transactions already in progress.

Security Updates

The `Security Checks` sections below contain details on the security checks performed during initialization of the SDK, as well as checks made periodically at runtime. The guidance in that section should be followed to make sure your application knows about these issues, and responds accordingly.

3DS SDK Flow

ActiveSDK can be used to support 3-D Secure V2 authentication natively in a mobile application. This SDK is built according to the 3DS V2 (2.1.0, 2.2.0) specification released by EMVCo. The EMV 3-D Secure specification, as well as more information on 3-D Secure V2 in general, can be found on the EMVCo website.

For a general overview of the sequence of 3DS SDK calls required to progress through 3-D Secure authentication, see the authentication sequence page.

Getting Started

The SDK binaries are available via the GPayments repository server. As part of the enrolment process, the credentials for the repository will be provided.

There are two versions of the 3DS SDK included with the installation: one to be used for development and one for production. The production version includes more strict security measures that would not allow for common development processes to occur, including running with attached debuggers or using simulators/emulators.

PCI 3DS SDK imposes requirements on 3DS SDKs that severely restrict development. For example, running on an emulator or debugging are forbidden in a production 3DS SDK. As such, the GPayments 3DS SDKs include both development and deployment versions, with the main differences being related to the security checks performed and resulting behavior. More details can be found in the Security Checks section.

Android

The following code snippet demonstrates how to add GPayments repository to the Android app.

> **Warning**
>
> The Android SDK is now built with AndroidX. Please ensure your project is also using AndroidX instead of the older Android Support Library. With Android Studio 3.2 and higher, you can migrate an existing project to AndroidX by selecting *Refactor -> Migrate to AndroidX* from the menu bar.

1. Put the following in the main `build.gradle` file, inside the `repositories` element:

Gradle

```
repositories {
    maven {
        url "https://repository.gpayments.com/repository/mvn-sdk"
        credentials {
            username = "put your username here"
            password = "put your password here"
        }
    }
}
```

2. Then add these lines inside the `dependencies` in the app's `build.gradle` file:

Production SDK Development SDK

```
dependencies {
   ...
   // Production SDK
   implementation 'com.gpayments.3ds2:activesdk-android:<version>@aar'
   ...
}
```

For the version number to use for this release, refer to the **SDK build version numbers** on the Release note page.

iOS

There are two ways of obtaining the SDK binaries for iOS, either through the GPayments repository server or via CocoaPods.

Downloading from the GPayments repository

Download the framework binary from the GPayments repository server and then follow these steps:

1. Place the framework directory into your project directory
2. In Xcode, click on your project. Then click on the target you want to link to. Finally, click on Build Phases.
3. Expand the Link Binary With Libraries section.
4. Click the +

5. Then click Add Other and choose Add Files
6. Navigate to framework directory
7. Click Open

Installing via CocoaPods

If you have not already installed CocoaPods, you can do so with the following command:

```
$ sudo gem install cocoapods
```

Then, navigate to the root of your project and run `$ pod init`

You will now see a generated Podfile in your root folder. This is where you can list all pods you wish to add to your project. Our currently available pods are listed below:

```
ActiveSDKDev
ActiveSDKProd
ActiveSDKSwift4.2Dev
ActiveSDKSwift4.2Prod
```

The pods are uploaded as an XCFramework for the versions of Swift that support it. An XCFramework is a single dependency for all target platforms and architectures, which means you do not need to specify the individual target platforms Device or Simulator separately anymore, as was the case in previous versions.

All pods without the specific version of Swift in their name are compiled with Swift 5. **ActiveSDKSwift4.2Dev** and **ActiveSDKSwift4.2Prod** are compiled using Swift 4.

In order to use an ActiveSDK pod, your podfile should be formatted like the following example:

```
source 'https://github.com/gpayments/activesdk-ios.git'

target 'ExampleProjectName' do
  pod 'ActiveSDKDevSimulator', '~> 2.2.7889'
end
```

For the version number to use for this release, refer to the **SDK build version numbers** on the Release note page.

> **Note**
>
> As the CocoaPod is hosted on GitHub, you can choose your preferred access method (HTTPS or SSH) for the CocoaPod repository. In order to use SSH, replace the first line of the podfile with `source 'git@github.com:gpayments/activesdk-ios.git'`.
>
> Before using SSH, you will need an SSH key linked to your GitHub account. If you do not have one yet, you can set it up using the guide from the GitHub Documentation.

Before running the next command you must ensure that you are logged into the correct Git account on the command line, which has been granted access to the GPayments private Git repository.

You can use any method for this, such as using Keychain Access in macOS, Git Credential Manager Core, or just ensuring that your Git configuration is set to the correct account. If you are using an account that has not been granted access to the private repository, the error `No `Podfile' found in the project directory.` will be thrown.

> **Important**
>
> If you do not already have your Git account credentials with access to the private repository, please contact your ActiveSDK provider.

Finally, run `$ pod install` in your root folder and the pod will be downloaded and integrated into your project.

Initializing the SDK

Depending on the 3DS Requestor application implementation, a `ThreeDS2Service` instance is created either during startup of the 3DS Requestor app (as a background task), or when a transaction is initiated. The state of the service is maintained until the `cleanup` method is called. To create an instance of the `ThreeDS2Service`, call the `initialize` method like so:

Android iOS

```
ThreeDS2Service.INSTANCE.initialize(applicationContext,
    configParameters,
    locale,
    uiCustomization,
    new MyClientEventListener(),
    new MySecurityEventListener());
```

The parameters this initialize method takes are as follows:

Android Parameters

Name	Type	Description
applicationContext	Context	An instance of the Android application Context. It can be the current activity instance, e.g. `MainActivity.this`.
configParameters	ConfigParameters	Configuration information that shall be used during initialization. For details, see the Config Parameters section.
locale	String	A string that represents the locale for the app's user interface (e.g. "en-US"). For future use.
uiCustomization	UiCustomization	A class that allows the customization of the 3DS SDK UI elements (fonts, colors, etc.). For details, see the UI Customization section.
clientEventListener	ClientEventListener	A listener to receive SDK events (for logging, etc.) For details, see the Client Event Listener section.
securityEventListener	SecurityEventListener	A listener to receive security warning events raised during SDK initialization and during runtime. For details, see the Security Event Listener section.

iOS Parameters

Name	Type	Description
configParameters	ConfigParameters	Configuration information that shall be used during initialization. For details, see the Config Parameters section.

Name	Type	Description
locale	String	A string that represents the locale for the app's user interface (e.g. "en-US"). For future use.
uiCustomization	UiCustomization	A class that allows the customization of the 3DS SDK UI elements (fonts, colors, etc.). For details, see the UI Customization section.
clientEventListener	ClientEventListener	A listener to receive SDK events (for logging, etc.) For details, see the Client Event Listener section.
securityEventListener	SecurityEventListener	A listener to receive security warning events raised during SDK initialization and during runtime. For details, see the Security Event Listener section.

Config Parameters

The `ConfigParameters` class is used to provide details required by the 3DS SDK for initialization, as well as additional optional settings. The `ConfigParameters.Builder` class helps with organizing these parameters which can be used as follows:

Android iOS

```
//adding the license
String runtimeLicense = ".....";


List<ConfigParameters.DirectoryServerInfo> directoryServerInfoList = new
ArrayList<>();
directoryServerInfoList.add(new
ConfigParameters.DirectoryServerInfo(DSInfo.DS_ID, KID, DSInfo.DS_CERT,
DSInfo.DS_CA_CERT));

List<String> deviceParameterBlacklist = new ArrayList<>();
    deviceParameterBlacklist.add("A009");
    deviceParameterBlacklist.add("A010");

List<String> maliciousApps = new ArrayList<>();
    maliciousApps.add("de.robv.android.xposed");

List<String> trustedAppStores = new ArrayList<>();
    trustedAppStores.add("com.xiaomi.market");

String appsignature = ParseAppSignature(HTTP.Get("https://www.example.com/
signature"));

List<String> clientConfigs = new ArrayList<>();
    clientConfigs.add("logLevel=3");
    clientConfigs.add("MaskSensitive=false");

ConfigParameters configParameters = new
ConfigParameters.Builder(directoryServerInfoList, runtimeLicense)
.deviceParameterBlacklist(deviceParameterBlacklist)
.maliciousApps(maliciousApps)
.trustedAppStores(trustedAppStores)
.appSignature(appsignature)
.clientConfig(clientConfigs)
.OOBAppURLSupported(true)
        .build();
```

Note

Information about the DSInfo class implementation can be found below.

Parameters fall into a few groups:

Android Parameters

Name	Type	Description
DirectoryServerInfo	List	Used to specify encryption keys and certificates for various supported direc details can be found in the Directory Server Info section.
RuntimeLicense	String	The 3DS SDK license string.
DeviceParameterBlacklist	List	A list of device parameters NOT to pull from the device. By default, the SDK device parameters as it can. This setting group can be used to instruct the S parameters. You can add parameters to this list by specifying the identifier. parameters can be found in the `EMV® 3-D Secure SDK Device Informatio` can be obtained from EMVCo directly.
MaliciousApps	List	Apps that are recognized as malicious. If the apps are installed, a security w raised. The following apps are considered malicious by default: `de.robv.an` `de.robv.android.xposed.installer`, `com.saurik.substrate`
TrustedAppStores	List	A security warning (SW02) will be raised if the app is not installed from a tru default, the Google Play Store (`com.android.vending`) is trusted. This para specify additional trusted app stores.
AppSignature	List	A security warning (SW02) is raised if this value does not match the app sig should be set to the SHA256 fingerprint of the certificate used to sign the ap specified, the SDK will see the signature as invalid. Note that this value shou in the app for security reasons. Instead, the fingerprint should be stored on a runtime, and set here.
ClientConfig	List	Configuration settings used for additional configuration of the SDK. A list of be found in the Client Config section.

After building the `ConfigParameters` object, the `addParams` method can be used to add additional parameters. This method accepts a group, parameter name, and parameter value. There are no specific groups associated with the SDK at this time, so the group can be set to a null value. For example:

```
configParameters.addParam(null, "ParameterName", "ParameterValue");
```

Available additional parameters are as follows:

Name	Description
ShowWhiteBoxInProcessingScreen	Display a white box behind the processing icon and directory server image. This takes a boolean string value ("True" or "False").
UseDefaultTrustedAppStoreList	This setting controls whether the default trusted app store list (i.e. com.android.vending) is used. This is true by default, but when set to false only those stores explicitly specified via the `TrustedAppStores` list during initialization will be trusted. This takes a boolean string value ("True" or "False").
ProgressBarColor	This setting controls the color of the progress bar displayed in the progress dialog returned by the `getProgressView` method. This accepts a hex color code value (e.g. "#b884f2").

iOS Parameters

Name	Type	Description
DirectoryServerInfo	List	Used to specify encryption keys and certificates for various supported directory servers. More details can be found in the Directory Server Info
RuntimeLicense	String	The 3DS SDK license string.
DeviceParameterBlacklist	List	A list of device parameters NOT to pull from the device. By default, the SDK will pull as many device parameters as it can. This setting group can be used to instruct the SDK not to pull certain parameters. You can add parameters to this list by specifying the identifier. A full list of parameters can be found in the `EMV® 3-D Secure SDK Device Information` document, which can be obtained from EMVCo directly.
ClientConfig	List	Configuration settings used for additional configuration of the SDK. A list of available options can be found in the Client Config section.
AppBundleID	String	The expected bundle identifier for the application. This should match the Bundle Identifier identity setting specified when building the application. A security warning is raised if this value does not match the Bundle ID of the application at runtime. If this value is not specified, the SDK will see the Bundle ID as invalid. Note that this value should not be hardcoded in the app for security reasons. Instead, the Bundle ID should be stored on a server, retrieved at runtime, and set here.

After building the `ConfigParameters` object, the `addParams` method can be used to add additional parameters. This method accepts a group, parameter name, and parameter value. There are no specific groups associated with the SDK at this time, so the group can be set to a null value. For example:

```
try configParameters.addParam(group: nil, paramName: "ParameterName", paramValue: "ParameterValue")
```

Available additional parameters are as follows:

Name	Description
ShowWhiteBoxInProcessingScreen	Display a white box behind the processing icon and directory server image. This takes a boolean string value ("True" or "False").
FireChallengeStatusFirst	This setting allows configuration of when the completed, cancelled, and protocolError Challenge Status Receiver events fire. By default ("False") these will fire after the challenge UI is closed. Setting this to "True" will cause these events to fire before the challenge UI closes.

UI Customization Options

The `UICustomization` class exposes functionality to customize the 3DS SDK UI elements. This allows the 3DS Requestor application to have its own look-and-feel in the native challenge pages. The customization of Buttons, Labels, TextBoxes, and Toolbars is enabled via the various classes and subclasses outlined below.

UI Customization

Main class for configuring the customization of UI elements.

Android iOS

```
public class UiCustomization {
    enum ButtonType (VERIFY, CONTINUE, NEXT, CANCEL, RESEND, ADDITIONAL,
OOB_OPEN_APP)
    enum LabelType (INFO_HEADER, INFO_TEXT, INFO_LABEL, WHITELIST, DEVICE_TYPE,
WHY_INFO, WHY_INFO_TEXT, EXPANDABLE_INFO, EXPANDABLE_INFO_TEXT, SELECTION_LIST,
DATA_ENTRY_LABEL, DATA_ENTRY_LABEL_2)

    void setButtonCustomization(ButtonCustomization buttonCustomization,
ButtonType buttonType)
    void setToolbarCustomization(ToolbarCustomization toolbarCustomization)
    void setLabelCustomization(LabelCustomization labelCustomization)
    void setTextBoxCustomization(TextBoxCustomization textboxCustomization)
    void setBackground(String hexColorCode)
    void setInformationZoneIconPosition(int position) // 0: right, 1: left, 2:
hide

    ButtonCustomization getButtonCustomization(ButtonType buttonType)
    ToolbarCustomization getToolbarCustomization()
    LabelCustomization getLabelCustomization()
    TextBoxCustomization getTextBoxCustomization()
    TextBoxCustomization getTextBoxTwoCustomization()
    String getBackground(String hexColorCode)
    int getInformationZoneIconPosition(int position)
}
```

Button Customization

The `ButtonCustomization` class enables the customization of various buttons presented to the cardholder during the challenge process.

Android iOS

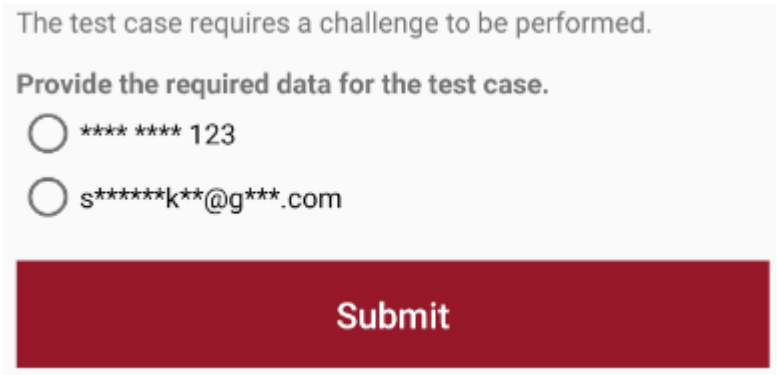
```
public class ButtonCustomization {
    void setBackgroundColor(String hexColorCode)
    void setCornerRadius(int cornerRadius)
    void setTextFontName(String fontName)
    void setTextFontSize(int fontSize)
    void setTextColor(String hexColorCode)
    void setHeight(int pixels)
    void setPadding(int start, int top, int end, int bottom) // applicable only
for ButtonType of CANCEL

    String getBackgroundColor()
    int getCornerRadius()
    String getTextFontName()
    int getTextFontSize()
    String getTextColor()
    int getHeight()
    int[] getPadding() // applicable only for ButtonType of CANCEL
}
```

For example, the following code will configure buttons with red backgrounds and white font:

Android iOS

```
UiCustomization uiCustomization = new UiCustomization();

ButtonCustomization customButton = new ButtonCustomization();
customButton.setTextColor("#FFFFFF"); // White
customButton.setBackgroundColor("#951728"); // Dark red

uiCustomization.setButtonCustomization(customButton,
UiCustomization.ButtonType.CANCEL);
uiCustomization.setButtonCustomization(customButton,
UiCustomization.ButtonType.CONTINUE);
uiCustomization.setButtonCustomization(customButton,
UiCustomization.ButtonType.NEXT);
uiCustomization.setButtonCustomization(customButton,
UiCustomization.ButtonType.RESEND);
uiCustomization.setButtonCustomization(customButton,
UiCustomization.ButtonType.SUBMIT);
```

This results in the following `Submit` button for a Single-Select challenge:



Label Customization

The `LabelCustomization` class enables the customization of various labels displayed throughout the challenge process.

Android iOS

```
public class LabelCustomization {
    void setHeadingTextAlignment(int textAlignment)
    void setHeadingTextColor(String hexColorCode)
    void setHeadingTextFontName(String fontName)
    void setHeadingTextFontSize(int fontSize)
    void setInputTextColor(String hexColorCode)
    void setInputTextFontName(String fontName)
    void setInputTextFontSize(int fontSize)
    void setTextColor(String hexColorCode)
    void setTextColor(LabelType labelType, String hexColorCode)
    void setTextFontName(String fontName)
    void setTextFontName(LabelType labelType, String fontName)
    void setTextFontSize(int fontSize)
    void setTextFontSize(LabelType, labelType, int fontSize)
    void setBackgroundColor(LabelType labelType, String hexColorCode)
    void setPadding(LabelType labelType, int start, int top, int end, int
bottom)

    int getHeadingTextAlignment()
    String getHeadingTextColor()
    String getHeadingTextFontName()
    int getHeadingTextFontSize()
    String getInputLabelTextColor()
    String getInputLabelTextFontName()
    int getInputLabelTextFontSize()
    String getTextColor()
    String getTextColor(LabelType labelType)
    String getTextFontName()
    String getTextFontName(LabelType labelType)
    int getTextFontSize()
    int getTextFontSize(LabelType labelType)
    String getBackgroundColor(LabelType labelType)
    int[] getPadding(LabelType labelType)
}
```

Text Box Customization

The `TextBoxCustomization` class enables the customization of text boxes displayed throughout the challenge process.

Android iOS

```
public class TextBoxCustomization {
    void setBorderColor(String hexColorCode)
    void setBorderWidth(int borderWidth)
    void setCornerRadius(int cornerRadius)
    void setTextColor(String hexColorCode)
    void setTextFontName(String fontName)
    void setTextFontSize(int fontSize)

    String getBorderColor()
    int getBorderWidth()
    int getCornerRadius()
    String getTextColor()
    String getTextFontName()
    int getTextFontSize()
}
```

Toolbar Customization

The `ToolbarCustomization` class enables the customization of the toolbar displayed throughout the challenge process.

Android iOS

```
public class ToolbarCustomization {
    void setBackgroundColor(String hexColorCode)
    void setButtonText(String buttonText)
    void setHeaderText(String headerText)
    void setTextColor(String hexColorCode)
    void setTextFontName(String fontName)
    void setTextFontSize(int fontSize)

    String getBackgroundColor()
    String getButtonText()
    String getHeaderText()
    String getTextColor()
    String getTextFontName()
    int getTextFontSize()
}
```

Icon Customization

iOS Only

Along with the customization of the look-and-feel of the UI via the UI Customization class detailed above, it may be desirable to also customize the images utilized by the SDK. There are seven images used by iOS for this that can be customized via a Bundle of images with the following names:

- **CollapsedLabelIndicator** The icon used to indicate a section that can be collapsed.
- **ExpandedLabelIndicator** The icon used to indicate a section that can be expanded.
- **MultiSelection** The icon used for a multi selection option.
- **MultiSelectionSelected** The icon used for selected multi selection option(s).
- **SingleSelection** The icon used for a single selection option.
- **SingleSelectionSelected** The icon used for the selected single selection option.
- **WarningIndicator:** The icon used (as indicated by the ACS) to draw attention to challenge information text being displayed.

The bundle containing these images can be passed to the ThreeDS2Service constructor like so:

```
let threeDS2Service = ThreeDS2Service(bundle: Bundle.main)
```

Dark Mode

The 3DS SDK supports Dark Mode based on settings configured at the system level. Nothing additional needs to be done in the code. Dark mode images can be configured for each of the above as well.

Client Event Listener

The `ClientEventListener` class exposes a number of events that are fired at certain points during the SDK's operation. Below is a list of the available events:

Data Packet In

Fired when receiving a data packet from the server.

Android iOS

```
public class MyClientEventListener implements ClientEventListener {
    ...
    public void fireDataPacketIn(byte[] dataPacket) {
        Log.i("ClientDataPacketIn", new String(dataPacket));
    }
    ...
}
```

This event fires when a packet is received. The entire data packet (including all framing and error detection characters) is contained in the `dataPacket` parameter. This parameter may be inspected for advanced troubleshooting, or to extract additional response properties beyond the scope of this component.

Data Packet Out

Fired when sending a data packet to the server.

Android iOS

```
public class MyClientEventListener implements ClientEventListener {
    ...
    public void fireDataPacketOut(byte[] dataPacket) {
        Log.i("ClientDataPacketOut", new String(dataPacket));
    }
    ...
}
```

This event fires right before each data packet is sent. The entire data packet (including all framing and error detection characters) is contained in the `dataPacket` parameter. This parameter may be inspected for advanced troubleshooting.

Log

Fires once for each log message.

Android iOS

```
public class MyClientEventListener implements ClientEventListener {
    ...
    public void fireLog(int logLevel, String message, String logType) {
        Log.i("ClientLog", logType + " - " + message);
    }
    ...
}
```

Logging is handled through the Log event. This will fire any time a message is built or a response is parsed, including error messages.

When the Log event is fired, the message in question is made available via the `message` event parameter. The other event arguments are `logType` and `logLevel`.

The `logType` parameter indicates the type of the log entry. Possible values are:

- Info
- RequestHeaders
- ResponseHeaders
- RequestBody
- ResponseBody
- ProxyRequest
- ProxyResponse
- FirewallRequest
- FirewallResponse
- CReq
- CRes
- Erro
- EphemeralKey
- DeviceParams
- SW

A `logType` of `SW` indicates a Security Warning. When this value is present, the `message` parameter will contain a code to indicate a more specific cause of the warning. Below is a list of possible warning `message` values and their meanings:

Android Security Warnings

SW Message Value	Description
01	Tampered; not installed from a trusted store
02	Tampered; internal error checking app store
03	Tampered; invalid appSignature
04	Tampered; suspicious app installed
05	Root; suspicious files present
06	Root; suspicious apk present
07	Root; root permissions
08	Root; root tag
09	Root; hooked
10	Debugging; debugger attached

iOS Security Warnings

SW Message Value	Description
01	Tampered, installed from Ad-Hoc or XCode
02	Tampered; invalid info dictionary
03	Tampered; Info.plist was modified after the app was built
04	Tampered; Bundle ID mismatch
05, XX	Jailbroken; suspicious jailbreak files present on the device

SW Message Value	Description
06, XX	Jailbroken; suspicious jailbreak library loaded
07, XX	Jailbroken; write permissions enabled on system directories
08	Jailbroken; app has permission to fork
09, XX	Jailbroken; app can query jailbreak URL schemes
10	Debugging; p_trace flag detected
11	Debugging; internal error: cannot find sysctl
12	Debugging; internal error: cannot find getpid
13	Debugging; invalid PID
14, XX	Debugging; reverse engineering tool detected on the device
15, XX	Debugging; suspicious TCP port detected

The LogLevel configuration setting can be used to specify the detail of the logs raised through the Log event. The `logLevel` parameter in the event indicates the log level to which the current message belongs. Possible values are:

- 0 - None
- 1 - Info
- 2 - Verbose
- 3 - Debug

Info level logs are available by default. The `LogLevel` configuration setting, specified in the Client Config settings, can be used to instruct the SDK to raise more or less detailed logs.

It is recommended to output all messages raised in this event to a file for record keeping or troubleshooting purposes.

SSL Server Authentication

Fired after the server presents its certificate to the client.

Android iOS

```
public class MyClientEventListener implements ClientEventListener {
    ...
    public void fireSSLServerAuthentication(byte[] certEncoded, String
certSubject, String certIssuer, String status, boolean[] accept) {
        //...
    }
    ...
}
```

This event fires when establishing a TLS connection and provides information about the server's certificate. The `accept` parameter indicates whether the certificate is trusted by default.

When `accept` is False, `status` shows why the verification failed (otherwise, `status` contains the string "OK"). If it is decided to continue, you can override and accept the certificate by setting the `accept` parameter to True.

SSL Status

Shows the progress of the secure connection.

Android iOS

```
public class MyClientEventListener implements ClientEventListener {
    ...
    public void fireSSLStatus(String message) {
        Log.i("ClientSSLStatus", message);
    }
    ...
}
```

The event is fired for informational and logging purposes only. Used to track the progress of the connection.

Security Event Listener

The `SecurityEventListener` class exposes a way to be notified of security issues that occur both during initialization of the SDK and at runtime. This is done via a single `alarm` event:

Alarm

Fired when a security issue is detected.

Android iOS

```
public class MySecurityEventListener implements SecurityEventListener {
    ...
    public void alarm(Severity severity, SecurityEvent event) {
        TransactionManager.getInstance().showToast("[Security Alarm] Severity: "
+ severity + ", Event: " + event);
    }
    ...
}
```

If a security issue is detected, the Alarm event will be raised. The severity of the error will be indicated by the `severity` event parameter. Possible values are:

- LOW
- MEDIUM
- HIGH

The `event` parameter will indicate which issue was encountered. Possible values are:

- ROOT
- TAMPERED
- INSTALLED_FROM_UNTRUSTED_STORE
- HOOK
- EMULATOR
- DEBUGGING
- DEBUG_ENABLED

For more information on the security checks performed by the component, and recommendations on handling detected issues, see the Security Checks section below.

Directory Server Info

`DirectoryServerInfo` parameters are used to specify certificate details for the various supported directory servers (DS). When creating a `Transaction`, the DS_ID is used to specify the

certificates to be used for that transaction. Each `DirectoryServerInfo` consists of three pieces of information:

- **DS_ID**: An identifier used internally in the SDK to match the transaction with a particular DS. Also referred to as a RID.
- **KID**: Key Identifier. There may be multiple keys for each DS with the keys each having an individual identifier.
- **DS_CERT**: Encryption certificate provided by the DS. Used to encrypt the DeviceInfo and generate keys to use for Challenge security.
- **DS_CA_CERT**: The root CA certificate used to issue the DS_CERT. This is used to verify the signature on the response from the DS (ARes packet), and can include the intermediate certificate as well.

The `DS_CERT` and `DS_CA_CERT` certificates are PEM encoded. When implementing the `DSInfo` class, the PEM encoded certificate content can be used as the static variable values. This is demonstrated with the example below for the GPayments TestLabs certificates:

```
public class DSInfo {

    public static final String DS_ID = "DS_RSA_TESTLAB_PROD";

    public static final String DS_CERT = "-----BEGIN CERTIFICATE-----\n" +

"MIIDyzCCArOgAwIBAgIUWcqKLMtnejuVbszZM00DLPZRqO4wDQYJKoZIhvcNAQEL\n" +

"BQAwdzELMAkGA1UEBhMCQVUxGDAWBgNVBAgMD05ldyBTb3V0aCBXYWxlczEPMA0G\n" +

"A1UEBwwGU3lkbmV5MRIwEAYDVQQKDAlHUGF5bWVudHMxEjAQBgNVBAsMCUFjdGl2\n" +

"ZVNESzEVMBMGA1UEAwwMQWN0aXZlU0RLLkNBMB4XDTIxMDcxNTA4MDMwOVoXDTIz\n" +

"MTAxODA4MDMwOVowcTELMAkGA1UEBhMCQVUxGDAWBgNVBAgMD05ldyBTb3V0aCBX\n" +

"YWxlczEPMA0GA1UEBwwGU3lkbmV5MRIwEAYDVQQKDAlBY3RpdmVTREsxEjAQBgNV\n" +

"BAsMCUFjdGl2ZVNESzEPMA0GA1UEAwwGU0RLIHYxMIIBIjANBgkqhkiG9w0BAQEF\n" +

"AAOCAQ8AMIIBCgKCAQEAzxcnpOaV9lGE4XWiafXtnYynlEMqrN65cqlovSd73nL4\n" +
                "WaAgl36lBVHb2/
vYEjIQLcf8VNme8bb9EvQmNulyva3ifBbQKRZT96XRVAzpj5qb\n" +

"47rpfskQ4gemR3XZZbqH2zMBPPTdXEKwkYbXPVE9riHkciLcbDO686zjsLLpTcmN\n" +
                "mqnFcJbSDggZfntKsTDro/
deenKKJd8q9yDlqO8KaZkXD1M2AWBF+aOEW5QINUg4\n" +
```

```
                "C4n89Q30Wva2c1q4S9DBczBHcSQntl9xIcOKrp3//
OJBP0Bl00S6CgxzpSnel1+J\n" +
                "en6cuDjbEm2WsOJrt5Nsjc+8LlYwIsuII3qd3Nuf/
QIDAQABo1UwUzAfBgNVHSME\n" +
                "GDAWgBTXgOf2gz2c4gvLTz6WbTL4/
QgKajAJBgNVHRMEAjAAMAsGA1UdDwQEAwIE\n" +

"8DAYBgNVHREEETAPgg1hY3RpdmVzZGsuY29tMA0GCSqGSIb3DQEBCwUAA4IBAQCx\n" +
                "8q1JOdZub+Obv/GABj/
eh4+3fSBdIuDQrCsNRC+uGWX7rfzWRQ0Px9qcK4cGbdVM\n" +

"SVek6zpF7m8LrX7AQBSXk6CgFWrjJqkwo8OzXddE4FTXdbBYKAIb5slhIXMVzPhu\n" +
                "3DnJ/QnXc2QfwrAaXSe/
vJ0eUNMm37dbJlGsLrJHFBWkZ9LphYg4Ji2TKYLIx5ek\n" +
                "3vxfj3iIFgCc/
RdAHKX38sG2+cASCKxonkK8SKgcqo5rKt+b4gny9yqSMEqfvT6j\n" +
                "4CyitN6ggL3nPnecc2MFpCSIYA+ZMf0UeLREOcVSg4o8rDB/
T8ZmGbDaz43E7AiB\n" +
                "sMcXGlVMEVh1isB/3SrW\n" +
                "-----END CERTIFICATE-----\n";

    public static final String DS_CA_CERT = "-----BEGIN CERTIFICATE-----\n" +
                "MIIDzzCCAregAwIBAgIUcfdJYkaK8cGVC8JX2RfP/
gWKHc4wDQYJKoZIhvcNAQEL\n" +

"BQAwdzELMAkGA1UEBhMCQVUxGDAWBgNVBAgMD05ldyBTb3V0aCBXYWxlczEPMA0G\n" +

"A1UEBwwGU3lkbmV5MRIwEAYDVQQKDAlHUGF5bWVudHMxEjAQBgNVBAsMCUFjdGl2\n" +

"ZVNESzEVMBMGA1UEAwwMQWN0aXZlU0RLLkNBMB4XDTIxMDcxNTA3NTc0NFoXDTI2\n" +

"MDcxNDA3NTc0NFowdzELMAkGA1UEBhMCQVUxGDAWBgNVBAgMD05ldyBTb3V0aCBX\n" +

"YWxlczEPMA0GA1UEBwwGU3lkbmV5MRIwEAYDVQQKDAlHUGF5bWVudHMxEjAQBgNV\n" +

"BAsMCUFjdGl2ZVNESzEVMBMGA1UEAwwMQWN0aXZlU0RLLkNBMIIBIjANBgkqhkiG\n" +
                "9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxG68SctQctVCM1L/
CIoFOr6W2TYbrx5Wj12Q\n" +
                "PTy/MWF0XZK8780iwI7xAjHeeyEMFPv/BOL6dzVtQYA/
3pNCdi5uhwqBhfvxeN0V\n" +
                "jo7A8zxFLH6/JkF4PUyffn10d3rvqKddOW/0/
gv1A32earnOVfhJympe1jfqgQbt\n" +

"Za8ryOxLVujP3OSyDL9hobTevd1ffYpnFyCHC4mqJVAELEoLEFWEp4ixlRh9yvUh\n" +

"j2icv4vHO6gE3MDWGjNoMwix3hsv6p1gQbzsUAWspg3UkjIALO3GnCY4U6Ml11jN\n" +

"cnp9cIP33madxY93vb0oKmJHCHsdBdVGLgcXFD2+gVAroN05pwIDAQABo1MwUTAd\n" +

"BgNVHQ4EFgQU14Dn9oM9nOILy08+lm0y+P0ICmowHwYDVR0jBBgwFoAU14Dn9oM9\n" +
                "nOILy08+lm0y+P0ICmowDwYDVR0TAQH/BAUwAwEB/
```

```
zANBgkqhkiG9w0BAQsFAAOC\n" +

"AQEAXtGL3CI9K2QugLv0HAPL2Jj0k9JMiV8RpQbMUVPmLQm9Tk48o3RKxZpCGizN\n" +
                "VKt3Lk5ryNAWkzypYfparZow8s/fRA0dXmZtq/
1Fjnj8ZJRmnYmKpIssdi+0NPr1\n" +
                "7V+2Y8g7wpinQ9w0tyMc7CVcqjlD3p+
+djiocjQ70HA6FdepxHF9Uj0Kzu71RecF\n" +
                "a7bqX9hdoS4dg9AkZzOQ6JF7NBcW225vhHJkQUekojW/
vloJPjnR7wDTaLrQVqq3\n" +
                "gIDwo2TgF/W1rIG9bf0hNx5V3fuOk1+Pv/8SE1tMnSj/KwHEeRKIWYrVn/
+sHKGQ\n" +
                "95DBV5KDrrNt5lr8CK5XhisHoQ==\n" +
                "-----END CERTIFICATE-----\n";

}
```

Using pre-configured Directory Server DS_ID's

In addition to adding the `DirectoryServerInfo` parameters above, the SDK comes pre-configured with `DirectoryServerInfo` for some card schemes production Directory Servers. Supported card schemes are:

Card scheme	DS_ID
Visa	A000000003
Mastercard	A000000004
Amex	A000000025
Discover	A000000152
JCB	A000000065
UnionPay	A000000333

To use the above pre-configured DS_DS's, just specify the DS_ID in the Creating a Transaction section. When using this pre-configured DS_ID no certificate is required for the `DirectoryServerInfo` parameter.

Important

While all care is taken to keep these pre-configred certificates up to date, it is always recommended that our customers continue to monitor and be aware of updates from card schemes in case changes do occur. In the case of certificate updates, a new version of the SDK will be made available. The update can be obtained from our site, but is not absolutely required since customers can manually update or add details in any version of the SDK (using the DirectoryServerInfoList parameter).

Client Config

The SDK accepts additional optional configuration settings which alter the behavior of the SDK. These are set as SETTING=VALUE pairs, and set as the `clientConfig` when building the `ConfigParameters`.

Log Level

This setting specifies the level of logging enabled in the component. Possible values include:

0 (None)	No events are logged.
1 (Info - default)	Informational events are logged.
2 (Verbose)	Detailed data is logged.
3 (Debug)	Debug data is logged.

This is set to 1 (Info) by default.

Mask Sensitive

This setting controls whether sensitive data is masked in the `Log` event. When set to `True` (default) the card number value will be replaced with the value `**********`.

Note: `DataPacketOut` will always contain the raw unmasked value regardless of this setting. This setting only applies to the `Log` event.

The default value is `True`.

Security Checks

The SDK performs security checks and collects device information during initialization. Warnings produced during the 3DS SDK initialization process can be retrieved using the `getWarnings` method, which returns a list of `Warning` objects. These can be checked by the app to determine whether to proceed with the checkout process or not:

Android iOS

```
/* check warnings */
List<Warning> warnings = ThreeDS2Service.INSTANCE.getWarnings();
if (warnings.size() > 0) {
    // process warnings
    // abort the checkout if necessary
}
```

The `Warning` object consists of the following fields:

Name	Type	Description
id	String	Warning identifier.
message	String	Warning message.
severity	Severity (enum)	Warning severity level (LOW, MEDIUM, HIGH).

The following warnings are documented in the EMVCo specification:

Security Warning ID	Description	Severity Level
SW01	The device is jailbroken.	HIGH
SW02	The integrity of the SDK has been tampered.	HIGH
SW03	An emulator is being used to run the app.	HIGH
SW04	A debugger is attached to the app.	MEDIUM
SW05	The OS or the OS version is not supported.	HIGH

To be notified of security issues both at initialization and at runtime, it is recommended to use the Security Event Listener as well.

Security Checks Performed During Initialization

The following security checks are performed during initialization of the SDK:

- Root detection: several ways of detecting if the device is rooted
- SDK tampering detection: including app signer fingerprint validation, hook detection, and malicious app detection
- Checks to make sure the app was installed from trusted app stores
- Emulator detection
- Detection of an attached debugger
- OS version detection

These initialization checks result in warnings accessible via the `getWarnings` detailed above. The message will contain a description of the issue encountered.

Security Checks Performed at Runtime

The following security checks are performed at runtime:

Android Security Checks

- Emulator detection (high severity): during `Transaction.getAuthenticationRequestParameters()`
- Debugging detection (medium severity): during `Transaction.doChallenge()`
- Root detection (high severity): during `Transaction.doChallenge()`
- Hook detection (high severity): during `ThreeDS2Service.createTransaction()`, and during the challenge process when the Submit/Resend/Cancel buttons are clicked
- Debug enabled detection (low severity): during `Transaction.doChallenge()`

iOS Security Checks

- Emulator detection (high severity): during `Transaction.doChallenge()`
- Debugging detection (medium severity): during `Transaction.doChallenge()`
- Method Swizzle Hook detection (high severity): during `Transaction.getAuthenticationRequestParameters()` and `Transaction.doChallenge()`

- Fishhook detection (high severity): during `Transaction.getAuthenticationRequestParameters()` and `Transaction.doChallenge()`
- Anti-debug (preventative): during `Transaction.getAuthenticationRequestParameters()`

These runtime checks result in firing of the Security Event Listener `alarm` event, and the activity closing for high severity issues. In the development version of the SDK (without `_deploy` in the file name), these checks are not performed.

Creating a Transaction

Call the `createTransaction` method to start a transaction. A reference should be kept for this transaction throughout the entire 3-D Secure process. This method takes the directory server ID, which points to a `DirectoryServerInfo` configuration parameter added during SDK initialization. The second parameter is a protocol version as a string. Supported version strings are `2.1.0` and `2.2.0`.

Android iOS

```
sdkTransaction = ThreeDS2Service.INSTANCE.createTransaction(DSInfo.DS_ID,
"2.2.0");
```

For information regarding the DS_ID, refer to the Directory Server Info section.

Later, when the transaction is complete, it should be closed via the Transaction object's `close` method.

Displaying Progress

A progress dialog must be displayed to the end user at certain points in the 3-D Secure process. While performing the challenge, the 3DS SDK will automatically display a progress dialog when appropriate.

A progress dialog must also be displayed to the user while the application is communicating with the 3DS Server and waiting for the authentication request results. The dialog used by the 3DS SDK is available to be used by the application if desired, and can be obtained from the `Transaction` object via the `getProgressView` method like so:

Android iOS

```
// The progress dialog is an instance of android.app.ProgressDialog
ProgressDialog sdkProgressDialog;

// Retrieve the progress dialog from the transaction object instance
sdkProgressDialog = sdkTransaction.getProgressView(mActivity);
sdkProgressDialog.show();

// Later, to hide/dismiss:
sdkProgressDialog.dismiss();
```

The 3DS SDK will automatically hide the dialog when starting the challenge process, so there is no need to explicitly dismiss the dialog in the application code.

Authenticating

The authentication process starts with a call to the `Transaction` object's `getAuthenticationRequestParameters` method. When this method is called, the 3DS SDK will encrypt the device information it collected during initialization and make it, along with other SDK information required by the 3DS Server, available in the returned `AuthenticationRequestParameters` object. This `AuthenticationRequestParameters` object consists of the following fields:

Name	Description
SDKTransactionId	A transaction identifier randomly generated by the SDK.
DeviceData	Device data collected by the SDK during initialization and encrypted using the DS key.
SDKEphemeralPublicKey	The public key component of the ephemeral key pair generated by the SDK.
SDKAppId	A UUID value, generated by the 3DS SDK, that uniquely identifies the app on the device.
SDKReferenceNumber	Assigned to our SDK by EMVCo after certification.
MessageVersion	From `createTransaction`, or a default value from the SDK (2.1.0).
AuthRequest	Packaged authentication request data to be sent to the 3DS Server.

The `AuthRequest` property is a packaged form of the other properties. The app would transmit this to the 3DS Server via a communication channel outside the scope of the 3DS SDK itself. Please refer to 3DS Requestor for the authentication message exchange implementation with ActiveServer. For other 3DS Server implmentations, please consult their documentation for details

Response Handling

The 3DS Server is responsible for reading the authentication response from the directory server and indicating to the client (in this case, the app) whether a challenge is required or not. For ActiveServer, check the `TransStatus` property after the authentication message exchange has been completed. If this is a value of `C` or `D`, a challenge is required.

If no challenge is required (frictionless flow), or authentication cannot be performed for some reason, no further action is required from a 3-D Secure standpoint. Proceed to the Completing Authentication section for details on closing the transaction.

If a challenge is required, the 3DS Server should return the required parameters to the app for use when starting the challenge process. The Performing a Challenge section details how to perform the challenge.

Performing a Challenge

If a challenge is required, the challenge is performed by creating the Challenge Parameters (using the authentication response from the 3DS Server) and following the steps in the Starting the Challenge section. The SDK will take care of all communication with the ACS while performing the challenge, as well as prompting the user as needed for the required input. When the challenge process is complete, the relevant Challenge Status Receiver callback will fire.

Challenge Status Receiver

`ChallengeStatusReceiver` is an interface used to obtain the result of the SDK challenge process. This should be implemented in the application and passed to the `doChallenge` method when Starting the Challenge. When the challenge process has finished, either successfully or in error, one of the following corresponding methods will be called:

Name	Description
completed	Called when the challenge process (that is, the transaction) is completed. When a transaction is completed, a transaction status shall be available.
cancelled	Called when the cardholder selects the option to cancel the transaction on the challenge screen.
timedout	Called when the challenge process reaches or exceeds the timeout interval that is specified during the `doChallenge` call.
protocolError	Called when the 3DS SDK receives an EMV 3-D Secure protocol-defined error message from the ACS.
runtimeError	Called when the 3DS SDK encounters errors during the challenge process. These errors include all errors except those covered by the `protocolError` method.

Example implementations of these methods can be found below:

Android iOS

```
@Override
public void completed(CompletionEvent completionEvent) {
    showToast("Challenge completed with transactionStatus " +
completionEvent.getTransactionStatus());
    closeTransaction();
}

@Override
public void cancelled() {
    showToast("Challenge cancelled.");
    closeTransaction();
}

@Override
public void timedout() {
    showToast("Challenge timed out.");
    closeTransaction();
}

@Override
public void protocolError(ProtocolErrorEvent protocolErrorEvent) {
    showToast("Challenge protocolError: " +
        protocolErrorEvent.getErrorMessage().getErrorDescription() + "\t" +
        protocolErrorEvent.getErrorMessage().getErrorDetails());
    closeTransaction();
}

@Override
public void runtimeError(RuntimeErrorEvent runtimeErrorEvent) {
    showToast("Challenge runtimeError: " + runtimeErrorEvent.getErrorMessage());
    closeTransaction();
}

// Helper method
public void closeTransaction() {
    if (sdkTransaction != null) {
        sdkTransaction.close();
        sdkTransaction = null;
    }
    sdkProgressDialog = null;
}
```

Challenge Parameters

The `ChallengeParameters` class holds parameters that are required to conduct the challenge process. This is passed to the `doChallenge` method when Starting the Challenge. Available properties, accessible via getters and setters, are:

Name	Type	Description
3DSServerTransactionID	String	3DS Server Transaction ID.
AcsTransactionID	String	ACS Transaction ID.
AcsRefNumber	String	ACS Reference Number.
AcsSignedContent	String	ACS signed content. This data includes the ACS URL, ACS ephemeral public key, and SDK ephemeral public key.
ThreeDSRequestorAppURL	String	3DS Requestor App URL.
ThreeDSServerAuthResponse	String	The Authentication Response from the 3DS Server component.

These properties can be set individually based on what comes back from the 3DS Server. If using the 3DS Server component, the `ThreeDSServerAuthResponse` can be set to the generated `ClientAuthResponse` value, and the other properties will be set automatically based on this value.

Android iOS

```
ChallengeParameters challengeParameters = new ChallengeParameters();
challengeParameters.setThreeDSServerAuthResponse(authResponse);
```

Starting the Challenge

If the response from the 3DS Server indicates that a challenge is required, the challenge process is initiated using the `doChallenge` method. When this is called, control of the UI is handed over to the 3DS SDK. The challenge will be performed by the SDK, and when finished the appropriate Challenge Status Receiver event will fire (completed, cancelled, etc.)

Android iOS

```
sdkTransaction.doChallenge(currentActivity, challengeParameters, this, 5);
```

The `doChallenge` method takes the following parameters:

Android Challenge Parameters

Name	Type	Description
currentActivity	android.app.Activity	The activity used by the SDK for the challenge process.
challengeParameters	ChallengeParameters	ACS details required by the 3DS SDK to conduct the challenge process during the transaction. For details, see the Challenge Parameters section.
challengeStatusReceiver	ChallengeStatusReceiver	Callback object for notifying the 3DS Requestor App about the challenge status. For details, see the Challenge Status Receiver section.
timeout	Integer	Timeout interval (in minutes) within which the challenge process must be completed. The minimum timeout interval is 5 minutes.

iOS Challenge Parameters

Name	Type	Description
navigationController	UIViewController	The view controller used by the SDK for the challenge process.
challengeParameters	ChallengeParameters	ACS details required by the 3DS SDK to conduct the challenge process during the transaction. For details, see the Challenge Parameters section.
challengeStatusReceiver	ChallengeStatusReceiver	Callback object for notifying the 3DS Requestor App about the challenge status. For details, see the Challenge Status Receiver section.

Name	Type	Description
timeout	Integer	Timeout interval (in minutes) within which the challenge process must be completed. The minimum timeout interval is 5 minutes.

Completing Authentication

If the `completed` callback fires, the `CompletionEvent` parameter will contain more details on the authentication results. This has two properties: `sdkTransactionID`, and `transactionStatus`. The `transactionStatus` contains the transStatus from the last CRes packet, which would be one of the values below:

- **Y:** Order completed successfully / Authentication successful
- **A:** Order completed successfully / Attempts processing performed
- **N:** Order denied / Not authenticated (denied)
- **R:** Order denied / Authentication rejected
- **U:** Order failed due to technical reasons / Authentication could not be performed

Managing Resources

The `cleanup` method frees up resources that are used by the 3DS SDK. It is called only once during a single 3DS Requestor App session:

Android iOS

```
ThreeDS2Service.INSTANCE.cleanup(applicationContext);
```

Get authentication result

For frictionless flow, the authentication result will return to the mobile App at the end of the authentication process. For challenge flow, a `transStatus=C` will be returned to the mobile App and a challenge process will be executed between Active SDK and the ACS. After the challenge is complete, for ActiveServer, the mobile App should call `getResult` endpoint to get the authentication result. Check the Requestor section for details of the `getResult` endpoint.

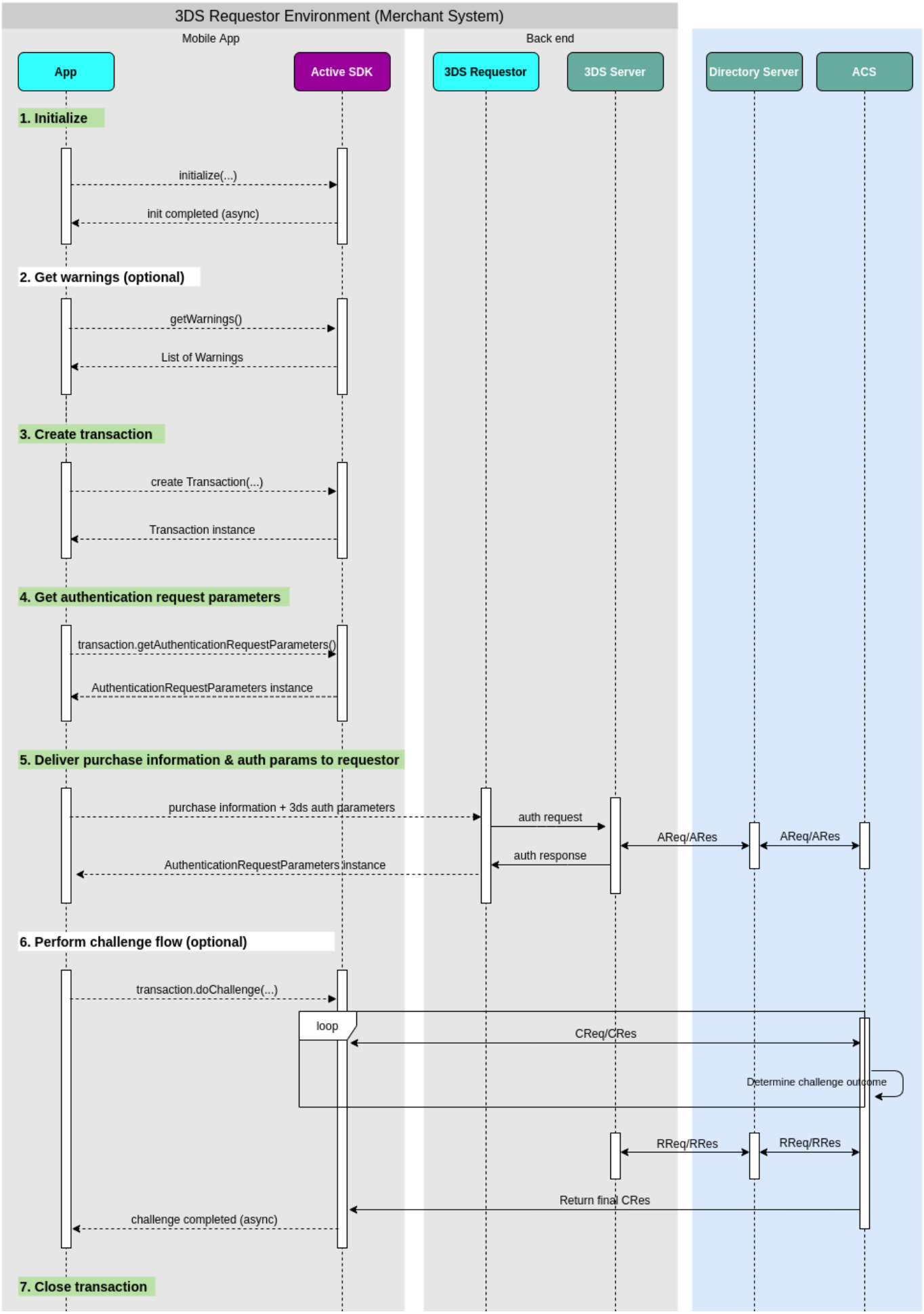
Application Permission Notes

The SDK itself does not require any specific permissions outside of internet access, and it will use only the permissions that the application has requested. The app should require permission before the SDK is initialized. This is required by the specification and some device parameters will not be available if permission is not granted.

For a full list of required permissions for each device parameter, see the EMV 3-D Secure SDK Device Information document, which can be obtained from EMVCo directly.

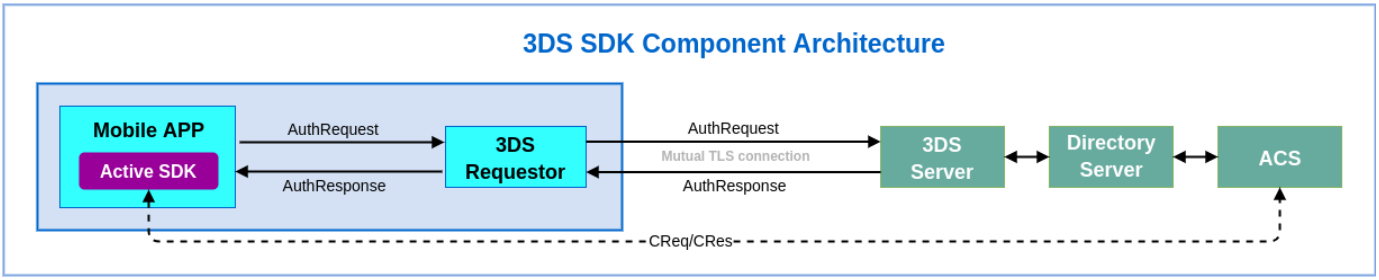
Authentication sequence

The following is a sequence diagram aimed to explain the concepts of the 3DS2 API interface, and visualize the interactions between each 3DS2 component.



3DS Requestor

To integrate ActiveSDK, the merchant site needs to implement a **3DS Requestor** in the backend. The following diagram shows the relationship between the ActiveSDK, the 3DS Requestor and other 3DS components. As defined by EMVCo's 3DSecure 2.0 specification, the communication between the merchant applications and the 3DS Server must be mutual authenticated. So a backend 3DS Requestor is needed to establish a mutual TLS connection with 3DS Server.



This set of documents will take you through the process of integrating the ActiveSDK with our 3DS Requestor demo code. In this demo, the 3DS Server which the 3DS Requestor connected to is our **ActiveServer**. You can modify the 3DS Requestor demo code to connect to any 3DS Servers.

The following is the prerequisite to using this guide:

- Knowledge of one of the following: Java, PHP, C#, or Go

Checkout and run the sample code

The 3DS Requestor demo code can be downloaded from the download page . Please follow the Get client certificate and Configure 3DS Requestor details instructions in the Requestor download page , and run the Requestor before moving onto next step.

> **Note**
>
> An activated and running ActiveServer instance (or a Saas ActiveServer instance) is required to run the Requestor demo code. Please contact GPayments if you need any help.

Implementation between Requestor and 3DS Server

The 3DS Requestor receives `AuthRequest` from the mobile APP and sends the request to 3DS Server. It also receives `AuthResponse` from 3DS Server and forwards the result to the mobile APP.

The demo **3DS Requestor** code provides the backend implementation with the following server side languages:

- **Java**: The java version is implemented based on the Springboot framework. For details of Springboot, check out https://spring.io/projects/spring-boot
- **C#**: The C# version is implemented based on ASP.net.
- **PHP**: The PHP version is implemented based on PHP 7.2 with cURL (Client URL Library).
- **Go**: The Go version is implemented based on Go 1.12 with Go Module support. All dependencies are listed in `go.mod` file.

Before starting the authentication process with 3DS Server, the 3DS Requestor needs to establish a mutual TLS connection with 3DS Server. Make sure you have followed the Get client certificate and Configure 3DS Requestor details instructions in the Requestor download page.

> **Tip**
>
> The implementation of TLS configuration for the HTTP Client can be found as follows:
>
> - Java: The TLS configuration and client certificate loading can be found in class `RestClientConfig.java`.
> - C#: The TLS configuration and client certificate loading can be found in class `RestClientHelper.cs`.
> - PHP: The TLS configuration and client certificate loading can be found in file `RestClientConfig.php`.
> - Go: The TLS configuration and client certificate loading can be found in file `https.go`.

Execute authentication

To execute authentication, the **3DS Requestor** needs to implement the `/v2/auth/app` endpoint to:

- Receive the `AuthRequest` from the mobile APP.
- Forward the request to 3DS Server.
- Receive the `AuthResponse` from 3DS Server.

- Return the response data to the mobile APP.

Java C# PHP Go

```
//AuthControllerV2.java
@PostMapping("/v2/auth/app")
@ResponseBody
public Message app(@RequestParam(value = "trans-type", required = false) String
transType,
    @RequestBody Message request) {
  return authServiceV2.app(transType, request);
}
//AuthServiceV2.java
public Message app(String transType, Message request) {
  //generate requestor trans ID
  String transId = UUID.randomUUID().toString();
  request.put(THREE_DS_REQUESTOR_TRANS_ID, transId);

  String appAuthUrl = config.getAsAuthUrl() + "/api/v2/auth/app";

  //Add parameter trans-type=prod in the appAuthUrl to use prod DS, otherwise
use TestLabs DS
  //For more details, refer to: https://docs.activeserver.cloud
  if ("prod".equals(transType)) {
    appAuthUrl = appAuthUrl + "?trans-type=prod";
  }

  logger.info("appAuthRequest on url: {}, body: \n{}", appAuthUrl, request);

  Message response =
      sendRequest(appAuthUrl, request, HttpMethod.POST);
  logger.info("appAuthResponse: \n{}", response);
  return response;
}
```

Tip

This demo code is to send `AuthRequest` to **ActivesServer**. To check the data structure of `AuthRequest` for ActiveServer, refer to the API document. To send `AuthRequest` to other 3DS Servers, please refer to the corresponding API documents.

ActiveServer users only

By default, the URL above will send the `AuthRequest` to GPayments TestLabs for testing purposes. When moving into production, to send the API request to the card scheme directory server, the *`trans-type` query parameter* must be appended to this API URL. See API description for further information on usage.

If you are using a Master Auth API client certificate to authenticate a **Business Admin** user on behalf of a merchant, the back-end needs to add a HTTP Header in the HTTP Request with a field of `AS-Merchant-Token` , which should be set to the merchantToken from the merchants profile. Detail implementation, refer to here.

Get authentication result

For frictionless flow, the authentication result will return to the mobile APP at the end of **Execute authentication** process.

For challenge flow, a `transStatus=C` will returned to the mobile APP at the end of **Execute authentication** process and a challenge process will be executed between ActiveSDK and ACS. After the challenge process finished, the mobile APP should call `/v2/auth/result` endpoint to get the authentication result. The 3DS Requestor needs to send a request to ActiveServer to get the result and return the result to the mobile APP.

Java C# PHP Go

```java
//AuthControllerV2.java
@ResponseBody
@GetMapping("/v2/auth/brw/result")
public Message resultBRW(@RequestParam("txid") String serverTransId) {
  return authServiceV2.getBRWResult(serverTransId);
}

//AuthServiceV2.java
public Message getBRWResult(String serverTransId) {
  //ActiveServer url for Retrieve Results
  String resultUrl = config.getAsAuthUrl() +
      "/api/v2/auth/brw/result?threeDSServerTransID=" +
      serverTransId;

  //Get authentication result from ActiveServer
  Message response = sendRequest(resultUrl, null, HttpMethod.GET);
  logger.info("authResponse: \n{}", response);

  return response;
}
```

Implementation between mobile App and Requestor

The mobile App needs to send the `AuthRequest` to Requestor. Following is a demo code of Android App to send the request. The iOS implementation is similar.

> **Warning**
>
> It is up to the merchant to implement the communication between the mobile App and the Requestor. It is required to use suitable authentication process to **secure the communication**.

Android

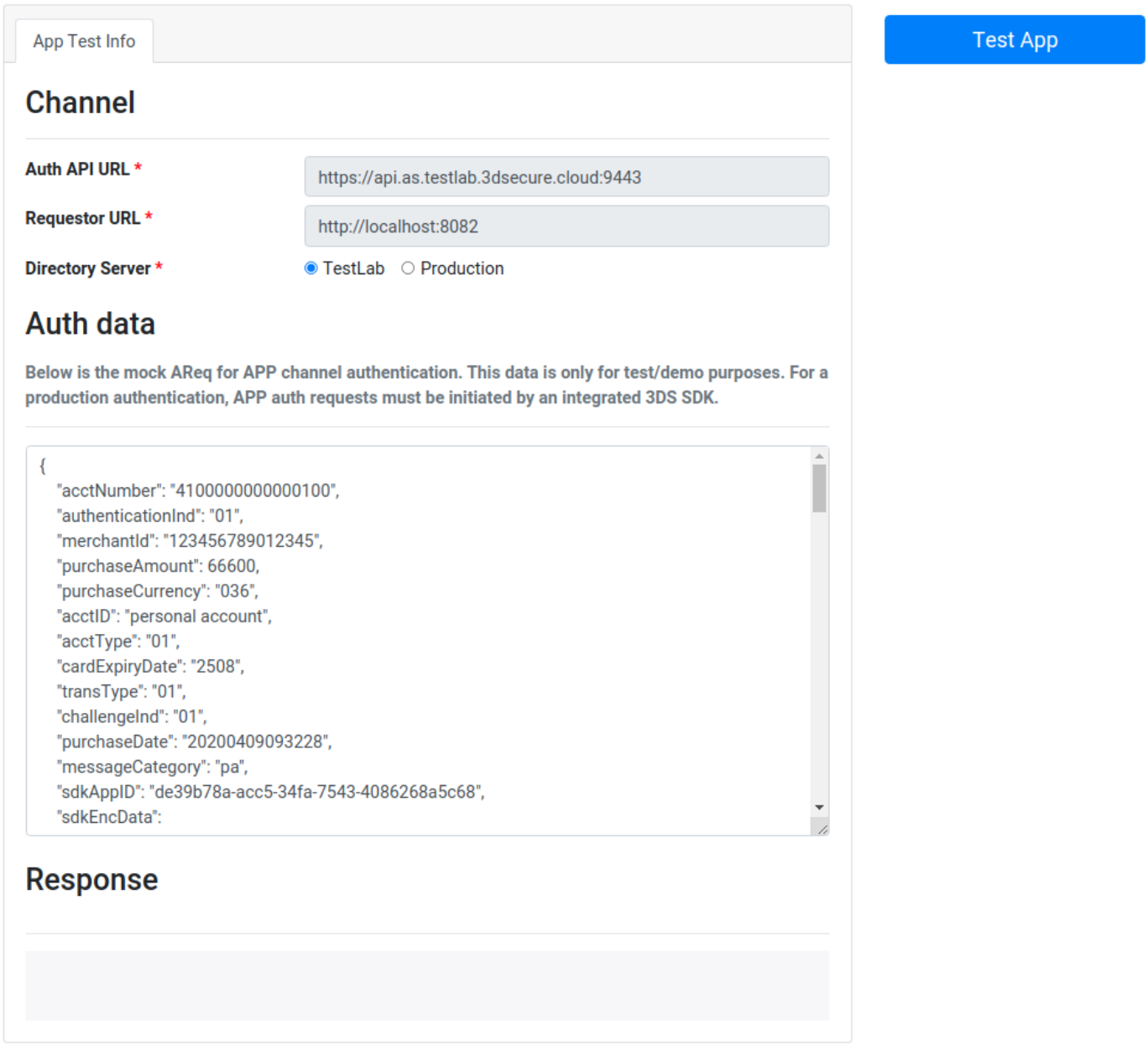
```
private Button purchaseBtn;
private OkHttpClient client = new OkHttpClient();
...
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  ...
  purchaseBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        ...
        AuthenticationRequestParameters authParams = transaction
            .getAuthenticationRequestParameters();

        PurchaseRequest appAuthReq = new PurchaseRequest(authParams,
            cardNumberEdit.getText().toString());

        final Request request = new Request.Builder()
            .url(REQUESTOR_URL + "/v2/auth/app")
            .post(RequestBody.create(JSON, appAuthReq.toJson()))
            .build();

        client.newCall(request).enqueue(new Callback() {
          ...
          @Override
          public void onResponse(Call call, Response response) throws
IOException {
            progress.dismiss();

            final String responseBody = response.body().string();
            Log.d(TAG, "Success purchase request: " + responseBody);
            ...
          }
        }
    }
}
```

Requestor troubleshooting

In the requestor, a front-end **App test** page is implemented for troubleshooting. Go to **App test** page by selecting `Test pages -> App`.

Mock `AuthRequest` data is pre-filled in this page, only for test/demo purposed. Press the `Test App` button, the mock data will be sent to the `/v2/auth/app endpoint` in the requestor back-

end. The `AuthResponse` will be shown on the `Response` area. Note, for a production authentication, `AuthRequest` must be initiated by an integrated 3DS SDK.



Glossary

This page provides a list of terms relating to 3D Secure 2.

Term	Definition
3DS Client	The consumer-facing component allowing consumer interaction with the 3DS Requestor for initiation of the EMV 3-D Secure protocol.
3DS Integrator	An EMV 3-D Secure participant that facilitates and integrates the 3DS Requestor Environment, and optionally facilitates integration between the Merchant and the Acquirer.
3DS Requestor	The initiator of the EMV 3-D Secure Authentication Request. For example, this may be a merchant or a digital wallet requesting authentication within a purchase flow.
3DS Requestor App	An App on a Consumer Device that can process a 3-D Secure transaction through the use of a 3DS SDK. The 3DS Requestor App is enabled through integration with the 3DS SDK.
3DS Requestor Environment	The 3DS Requestor-controlled components (3DS Requestor App, 3DS SDK, and 3DS Server) are typically facilitated by the 3DS Integrator. Implementation of the 3DS Requestor Environment will vary as defined by the 3DS Integrator.
3DS SDK	3-D Secure Software Development Kit (SDK). A component that is incorporated into the 3DS Requestor App. The 3DS SDK performs functions related to 3-D Secure on behalf of the 3DS Server.
3DS Server	Refers to the 3DS Integrator's server or systems that handle online transactions and facilitates communication between the 3DS Requestor and the DS.
3-D Secure (3DS)	An e-commerce authentication protocol that enables the secure processing of payment, non-payment and account confirmation card transactions.
Access Control Server (ACS)	A component that operates in the Issuer Domain, that verifies whether authentication is available for a card number and device type, and authenticates specific Cardholders.
Authentication	In the context of 3-D Secure, the process of confirming that the person making an e-commerce transaction is entitled to use the payment card.

Term	Definition
Authentication Request(AReq) Message	An EMV 3-D Secure message sent by the 3DS Server via the DS to theACS to initiate the authentication process.
Authentication Response (ARes) Message	An EMV 3-D Secure message returned by the ACS via the DS in response to an Authentication Request message.
Authentication Value(AV)	A cryptographic value generated by the ACS to provide a way, during authorisation processing, for the authorisation system to validate the integrity of the authentication result. The AV algorithm is defined by each Payment System.
Authorisation	A process by which an Issuer, or a processor on the Issuer's behalf, approves a transaction for payment.
Authorisation	System The systems and services through which a Payment System delivers online financial processing, authorisation, clearing, and settlement services to Issuers and Acquirers.
Bank Identification Number (BIN)	The first six digits of a payment card account number that uniquely identifies the issuing financial institution. Also referred to as Issuer Identification Number (IIN) in ISO 7812.
Base64	Encoding applied to the Authentication Value data element as defined in RFC 2045.
Base64url	Encoding applied to the 3DS Method Data, Device Information and the CReq/CRes messages as defined in RFC 7515.
Card	In EMVCo Core specification, synonymous to the account of a payment card.
Cardholder	An individual to whom a card is issued or who is authorised to use that card.
Challenge	The process where the ACS is in communication with the 3DS Client to obtain additional information through Cardholder interaction.
Challenge Flow	A 3-D Secure flow that involves Cardholder interaction as defined in EMVCo Core Spec Section 2.5.2.
Challenge Request(CReq) Message	An EMV 3-D Secure message sent by the 3DS SDK or 3DS Server where additional information is sent from the Cardholder to the ACS to support the authentication process.

Term	Definition
Challenge Response(CRes)	The ACS response to the CReq message. It can indicate the result of the Cardholder authentication or, in the case of an App-based model, also signal that further Cardholder interaction is required to complete the authentication.
Consumer Device	Device used by a Cardholder such as a smartphone, laptop, or tablet that the Cardholder uses to conduct payment activities including authentication and purchase.
Device Channel	Indicates the channel from which the transaction originated. Either: • App-based (01-APP) • Browser-based (02-BRW) • 3DS Requestor Initiated (03-3RI)
Device Information	Data provided by the Consumer Device that is used in the authentication process.
Directory Server (DS)	A server component operated in the Interoperability Domain; it performs a number of functions that include: authenticating the 3DS Server, routing messages between the 3DS Server and the ACS, and validating the 3DS Server, the 3DS SDK, and the 3DS Requestor.
Directory Server Certificate Authority (DS CA)	A component that operates in the Interoperability Domain; generates and Certificate Authority (DS distributes selected digital certificates to components participating in 3-D Secure. Typically, the Payment System to which the DS is connected operates the CA.
Directory Server ID (directoryServerID)	Registered Application Provider Identifier (RID) that is unique to the Payment System. RIDs are defined by the ISO 7816-5 standard.
Electronic Commerce Indicator (ECI)	Payment System-specific value provided by the ACS to indicate the results of the attempt to authenticate the Cardholder.
Frictionless	The process of authentication achieved without Cardholder interaction.
Frictionless Flow	A 3-D Secure flow that does not involve Cardholder interaction as defined in EMVCo Core Spec Section 2.5.1.
Merchant	Entity that contracts with an Acquirer to accept payment cards. Manages the online shopping experience with the Cardholder, obtains card number, and then transfers control to the 3DS Server, which conducts payment authentication.
One-Time Passcode (OTP)	A passcode that is valid for only one login session or transaction, on a computer system or other digital device.

Term	Definition
Out-of-Band (OOB)	A Challenge activity that is completed outside of, but in parallel to, the 3-D Secure flow. The final Challenge Request is not used to carry the data to be checked by the ACS but signals only that the authentication has been completed. ACS authentication methods or implementations are not defined by the 3-D Secure specification.
Registered Application Provider Identifier (RID)	Registered Application Provider Identifier (RID) is unique to a Payment System. RIDs are defined by the ISO 7816-5 standard and are issued by the ISO/IEC 7816-5 registration authority. RIDs are 5 bytes.

Support

About GPayments

Who are we

GPayments is an Australian company that specialises in delivering payment authentication products for online transactions, and provides a range of solutions for card schemes, financial institutions (both issuers and acquirers), online service providers, merchants, as well as cardholders in many countries around the world. With over fifteen years' experience, GPayments has a long history of pioneering and innovating within the industry, and has positioned itself as a leader in secure online commerce.

Our mission

We are focused on providing robust payment and authentication solutions for online transactions. We provide services for financial institutions (both issuers and acquirers), service providers, merchants and cardholders. With over two decades of experience in technology development, we have established ourselves as a leader in secure electronic commerce.

What we do

We provide a complete range of integrated authentication products based on the 3-D Secure protocol, Verified by Visa, Mastercard SecureCode, JCB J/Secure, American Express SafeKey and Diners Club International ProtectBuy. These products are deployed in over 30 countries worldwide and in some of the largest international corporations.

Why choose us

With our proven track record in providing open, scalable solutions which include multi-currency and multilingual support for global markets, we are innovators in the payment authentication industry. We constantly examine and evaluate trends in the online payments industry in efforts to evolve and revolutionise. We also provide systems integration and customer support via our vast partnership network.

For more information about GPayments, please visit our website at https://www.gpayments.com/.

GPayments
a neurocom company

Language | Login

Our Solutions | Resources | About Us | Contact Us

Frictionless Fraud Prevention Simplified

Securing the payment ecosystem end-to-end with fraud and risk management solutions for financial institutions, payment service providers and merchants.

Explore products | Get in touch

AS SEEN ON
FINANCIAL REVIEW

Robust cardholder authentication solutions for online transactions

We provide a complete range of integrated authentication products based on the 3D Secure protocol, Visa Secure, Mastercard Identity Check, JCB J/Secure, American Express SafeKey, UnionPay International 3-D Secure and Discover ProtectBuy.

bambora | ERSTE | OKU | Paynova | UPC | zalando | SecurePay | DS Corporation

Solutions built to help

Payment Gateways & Merchants

- Decrease cart abandonment rates with 3DS2's Frictionless Flow and mobile SDK
- Reduce credit card fraud and eliminate chargebacks via liability shift
- Comply with local regulations, such as PSD2

See Related Products

Card Issuers

- Stay in control of authenticating transactions
- Collects a rich data set for high authentication accuracy
- Reduce credit card fraud

See Related Products

Contact Us

If you find any errors in the documentation, or would like to contact us for additional support, please email GPayments Tech Support at techsupport@gpayments.com.

Document Control

Date	ActiveSDK Version	Documentation Version	Change Details
16/04/24	**Android**: 2.3.8845 **iOS**: 2.3.8845	2.3.1:1	• Updated the Release notes
22/12/23	**Android**: 2.3.8718 **iOS**: 2.3.8738	2.3.0:1	• Updated the Release notes • Updated Config Parameters section • Updated UI Customization section
17/05/23	**Android**: 2.2.8510 **iOS**: 2.2.8510	2.2.7:1	• Updated the Release notes
30/08/22	**Android**: 2.2.8237 **iOS**: 2.2.8237	2.2.6:1	• Updated the Release notes
17/02/22	**Android**: 2.2.8070 **iOS**: 2.2.8054	2.2.5:1	• Updated the Release notes
03/02/22	**Android**: 2.2.8054 **iOS**: 2.2.8054	2.2.4:1	• Added a note to use AndroidX to the Getting Started • Updated the sample code example Android and iOS in the Challenge Status Receiver • Added **ThreeDSServerAuthResponse** to the table in Challenge Parameters and changed the sample code

Date	ActiveSDK Version	Documentation Version	Change Details
24/11/21	**Android**: 2.2.7989 **iOS**: 2.2.7989	2.2.3:1	• Pre-configured 3DS SDK certificate values have been added/updated for AMEX, Discover, JCB and UnionPay in the Using pre-configured Directory Server DS_ID's table • Uploaded the pods as an XCFramework for the versions of Swift that support it. More information can be found at Installing via CocoaPods
15/10/21	**Android**: 2.2.7936 **iOS**: 2.2.7936	2.2.2:2	• Changed the sample code example for Android **applicationContext** in the Integration guide
01/10/21	**Android**: 2.2.7936 **iOS**: 2.2.7936	2.2.2:1	• Added a section on using pre-configured directory server DS_ID's to the Integration guide • Added example repository link for Android Dev SDK to the Integration guide • Added a note regarding CocoaPod repository access method to the Integration guide • Added an example for adding DS Certificates to the Directory Server Info on the Integration Guide • Added the Release notes page
20/09/21	**Android**: 2.2.7722 **iOS**: 2.2.7876	2.2.1:3	• Removed Android Dev SDK instructions from the Integration guide
20/08/21	**Android**: 2.2.7722 **iOS**: 2.2.7876	2.2.1:2	• Added instructions for CocoaPod integration • Added the Document Control
30/07/21	**Android**: 2.2.7722 **iOS**: 2.2.7876	2.2.1:1	• Initial release

Release notes

ActiveSDK v2.3.1

[Release Date: 16/04/2024]

SDK build version numbers (for repository use):

- Android: 2.3.8845
- iOS: 2.3.8845

Change	Description
Certificate update	Mastercard issued a new encryption certificate. The previous encryption certificate was set to expire on June 15, 2024; the new certificate expires on July 15, 2026.
Certificate update	Visa issued a new encryption certificate. The previous encryption certificate was set to expire on August 22, 2024; the new certificate expires on February 26, 2027.

ActiveSDK v2.3.0

[Release Date: 22/12/2023]

SDK build version numbers (for repository use):

- Android: 2.3.8718
- iOS: 2.3.8738

Change	Description
DirectoryServerInfo	Added new KID parameter to DirectoryServerInfo
ConfigParameters.Builder	Added new OOBAppURLSupported method to ConfigParameters.Builder
UiCustomization.ButtonType	Added new enum values ADDITIONAL and OOB_OPEN_APP to UiCustomization.ButtonType

Change	Description
UiCustomization.LabelType	Added new enum values DEVICE_BINDING, DATA_ENTRY_LABEL and DATA_ENTRY_LABEL_2 to UiCustomization.LabelType
UiCustomization	Added new getTextBoxTwoCustomization method to UiCustomization

ActiveSDK v2.2.8

[Release Date: 19/07/2023]

SDK build version numbers (for repository use):

- Android: 2.2.8593
- iOS: 2.2.8593

Change	Description
OOB (Android)	Added the required BROWSABLE category to the manifest for OOB challenge listener intent filter.
R8 compiler (Android)	Fixed issue related to synchronization when using the R8 compiler.
[**IMPORTANT**] American Express	New encryption certificate, which is used by the SDK to encrypt the device information. The previous encryption certificate was set to expire on August 18th, 2023.
Xcode (iOS)	Upgraded the iOS environment to Xcode 14. Apple deprecated building iOS projects with deployment targets for the arm7 and i386 architectures.

ActiveSDK v2.2.7

[Release Date: 17/05/2023]

SDK build version numbers (for repository use):

- Android: 2.2.8510
- iOS: 2.2.8510

Change	Description
Challenge flow (Android)	Fixed issue where SDK would crash when trying to dismiss activity during a challenge.
Improvement (Android)	Add support for Parcelable interface.
Version support	Added support for Android 13.
App URL (iOS/ Android)	Fixed issue where the ThreeDSRequestorAppURL was not added to 2.2.0 CReq messages.
OOB flow (Android)	Fixed issue for OOB flow where the CReq was not automatically posted when the requestor app was moved to the foreground.

ActiveSDK v2.2.6

[Release Date: 30/08/2022]

SDK build version numbers (for repository use):

- Android: 2.2.8237
- iOS: 2.2.8237

Change	Description
Certificate update	Pre-configured 3DS SDK encryption certificate value has been updated for UnionPay.
ProGuard (Android)	Added configuration file so no obfuscation rules would need to be manually specified by users that also use obfuscation.
Bluetooth Permission (Android)	Fixed a bug where the BLUETOOTH_CONNECT permission was required for Android 12 and up. If this permission has not been granted, device parameter C009 will not be collected and will instead be listed in the Device Parameter Not Available section with a value of RE03 (indicating permissions not granted).

ActiveSDK v2.2.5

[Release Date: 17/02/2022]

SDK build version numbers (for repository use):

- Android: 2.2.8070
- iOS: 2.2.8054

Change	Description
Challenge Message (Android)	Fixed challengeDataEntry and challengeHTMLDataEntry formatting in CReq.

ActiveSDK v2.2.4

[Release Date: 03/02/2022]

SDK build version numbers (for repository use):

- Android: 2.2.8054
- iOS: 2.2.8054

Change	Description
AndroidX	The Android SDK is now built with AndroidX which replaces the original Android Support Library.
Challenge UI (iOS)	Fixed an issue where the Challenge UI is not closed when the close method is called in protocolError.
Challenge UI (iOS)	Fixed an issue where both cancel and protocolError fire if the ACS returns an error during cancel. In this case, only protocolError should fire.
Challenge UI (Android)	Fixed an issue where the Challenge UI is not closed if there is a protocolError when clicking the resend button.
Fix	Implemented SDKMaxTimeout. SDK will now close the transaction UI when a timeout occurs.

ActiveSDK v2.2.3

[Release Date: 24/11/2021]

SDK build version numbers (for repository use):

- Android: 2.2.7989
- iOS: 2.2.7989

Change	Description
Certificate Update	Updated the pre-configured certificates for the AMEX, Discover, JCB and UnionPay Directory Servers
New Format of Packaging	An XCFramework is a single dependency for all target platforms and architectures, which means you do not need to specify the individual target platforms Device or Simulator separately
Activity leak (android only)	When cleanup was called, the resources were not being released properly which caused a memory leak and a RuntimeException error. This has been resolved
Encoding bug	There was an issue with encoding of Unicode data in the SDK, and this has been resolved

ActiveSDK v2.2.2

[Release Date: 01/10/2021]

SDK build version numbers (for repository use):

- Android: 2.2.7936
- iOS: 2.2.7936

Change	Description
Certificate Update	Updated the pre-configured certificates for the Visa and Mastercard Directory Servers

ActiveSDK v2.2.1

[Release Date: 30/07/2021]

SDK build version numbers (for repository use):

- Android: 2.2.7722
- iOS: 2.2.7876

Change	Description
Release	Initial release